Segmentation and 3D reconstruction of rose plants from stereoscopic images

Segmentation and 3D reconstruction of rose plants from stereoscopic images

Hanz Cuevas-Velasquez[a,*], Antonio-Javier Gallego[b], Robert B. Fisher[a]

[a] *School of Informatics, University of Edinburgh, Edinburgh, EH8 9AB, UK*
[b] *Department of Software and Computing Systems, University of Alicante, 03690 Alicante, Spain*

Abstract

The method proposed in this paper is part of the vision module of a garden robot capable of navigating towards rose bushes and clip them according to a set of pruning rules. The method is responsible for performing the segmentation of the branches and recovering their morphology in 3D. The obtained reconstruction allows the manipulator of the robot to select the candidate branches to be pruned. This method first obtains a stereo pair of images and calculates the disparity image using block matching and the segmentation of the branches using a Fully Convolutional Neuronal Network modified to return a map with the probability at the pixel level of the presence of a branch. A post-processing step combines the segmentation and the disparity in order to improve the results. Then, the skeleton of the plant and the branching structure are calculated, and finally, the 3D reconstruction is obtained. The proposed approach is evaluated with five different datasets, three of them compiled by the authors and two from the state of the art, including indoor and outdoor scenes with uncontrolled environments. The different steps of the proposed pipeline are evaluated and compared with other state-of-the-art methods, showing that the accuracy of the segmentation improves other methods for this task, even with variable lighting, and also that the skeletonization and the reconstruction processes obtain robust results.

Keywords: Computer Vision, Stereo Vision, Semantic Segmentation, 3D modelling, Automated agriculture

1. Introduction

Computer vision and robotics have made significant advances in detection and automation of indoor and outdoor tasks. The vision part is generally used to segment, localize or track objects so the robot can navigate to an area of interest and manipulate the object [16]. Outdoor tasks usually deal with a more uncontrolled environment than indoor tasks, mainly due to the variations of light, wind, shadows, as well as variations in the type of terrain. An example of a challenging outdoor task is that of a robot that can move through a garden, detect key elements and recover their structure in order to work with them.

The method proposed in this paper belongs to the vision module of a garden robot capable of navigating towards rose bushes and clip them according to a set of pruning rules[1]. This robot (see Figure 1) consists of a mobile platform with a camera rig for navigation and a 6-DOF robotic arm with a cutting tool and a stereo camera mounted on the end-effector using an eye-in-hand configuration [29]. A detailed description of the robot can be found in [56]. A demonstration of the rose pruning process can be also seen in the video: `https://youtu.be/r9IHy5lH8YM`.

*Corresponding author: Email: hanz.c.v@ed.ac.uk
Email addresses: `hanz.c.v@ed.ac.uk` (Hanz Cuevas-Velasquez), `jgallego@dlsi.ua.es` (Antonio-Javier Gallego), `rbf@inf.ed.ac.uk` (Robert B. Fisher)

[1] Project TrimBot2020: `http://trimbot2020.webhosting.rug.nl`

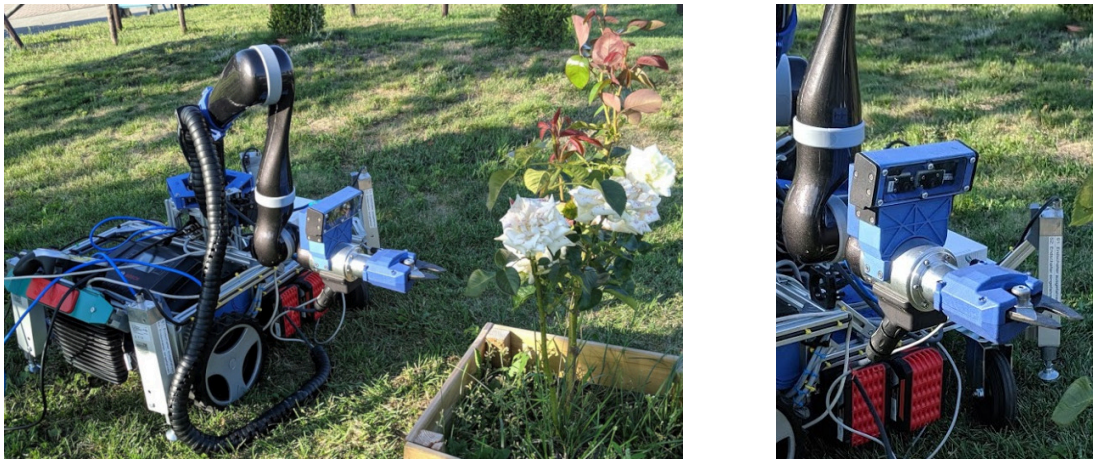


Figure 1: General overview of the robot (left image) including the 6-DOF robotic arm and the cutting tool. A stereo camera using an eye-in-hand configuration (right image) is mounted on the same cutting tool.

The vision module is divided in two main parts, one for robot navigation and other for visual servoing. The robot navigation uses a depth fusion system which combines multiple disparity images obtained from a 10 camera rig [43] and SLAM for robot localization [48]. This part allows the robot to navigate in the garden towards a rose bush. The second part of the vision module implements visual servoing for the manipulator. This is in charge of processing the images captured by a stereo camera mounted on the robot arm to detect cutting locations on a rose branch and move the cutter towards those cutting locations.

The first module allows the robot to navigate towards a rose bush and, once a rose bush is located and the robot is facing towards it, the system switches to the second module. This second module assumes it will receive images where only one rose bush appears and all the segmented branches belong to the same plant.

The pruning process is: First, the branches are localized by segmenting them from the rest of the scene. Then, the 3D morphology of the plant is recovered. Finally, based on the morphology of the plant, the robot finds the branches that should be cut and sends the manipulator to clip them. This paper focuses on the vision module, which segments the stems of a rose bush and obtains its morphology. To the best of our knowledge, there are no previous methods devoted to segment roses and obtain their branching structure for pruning. The closest approaches are the segmentation of trees and plants, which are described in detail in the following section.

Our approach tries to solve this problem without making assumptions about the type of environment or lighting conditions, working with aligned stereo input images and recovering the morphology of the plant to determine the branches to prune. First, a Selectional Autoencoder architecture [12] is trained to select the pixels that belong to the branches of a rose bush. It then calculates the disparity map and combines it with segmentation to improve the accuracy of the result. Finally, the skeleton of the bush, the branches and the 3D morphology of the plant are obtained.

This approach is validated using five different datasets: (1) A synthetic dataset of rose bushes in 4 different environments, (2) a dataset of realistic plastic rose bushes with real backgrounds, (3) a dataset of real roses captured in two different botanic gardens, (4) a dataset from the state of the art of *Arabidopsis* genus plants captured indoors, and (5) another dataset from the state of the art of real roses captured in a real garden.

In summary, **the paper makes the following contributions**: (1) Rose stem segmentation through Selectional Autoencoders adjusted to work on real environments with variable light conditions, (2) novel combination of binary segmentation with disparity to obtain the 3D morphology of the plant through skeletonization, (3) a complete pipeline to recover the morphology of rose stems from stereo images that allows to determine the 3D structure of branches and choose the ones that should be pruned, and (4) a collection of datasets for rose bush segmentation.

The rest of the paper is organized as follows: the next section makes a brief review of the state of the art, Section 3 presents the proposed approach, Section 4 describes the datasets used in the evaluation, Section 5 reports the evaluation results, and finally, conclusions and future work are addressed in Section 6.

2. Related work

As stated before, there is no literature on the specific task of rose bush segmentation and morphology extraction using 2D stereo images. Therefore, we will focus on works that are closely related, mainly in the areas of tree and plant modeling, either using 2D images, multiple 2D images or point clouds. There is an extensive literature on these topics, so we will review it by grouping them according to the type of data used.

Among the methods that use point clouds we can find works that only reconstruct the branches (without leaves, called off-leaf) and others that do include them (on-leaf). For example, Raumonen et al. [45] model off-leaves trees by fitting cylinders to the point cloud. Then, they find the tree components by considering small regions of the tree and searching for neighbors and bifurcations. Hackenberg et al. [26] propose a similar approach but using spheres to look for sections of the branch instead of splitting the tree in regions. These two approaches obtain the plant information using point clouds of off-leaf trees. However, they need a clean point cloud as input while our approach uses a raw stereo image as a starting point. There are also works on getting the morphology of trees with leaves. Livny et al. [38] propose an on-leaf model to reconstruct the skeletal structures of trees from a point cloud. They perform global optimization and clustering on a directed acyclic graph that represents the points of a tree. Belton et al. [7] obtain geometric features and use Gaussian Mixture Models (GMM) to split the tree into its principal parts (trunk, branches and leaves). Huang et al. [30] get the skeleton of a plant by applying *L1 - medial skeleton* on a point clouds. While this method works well to obtain the 3D skeleton of any 3D shape. It does not capture any morphological information of the plant like our method. Tabb & Medeiros [57] go a step further and not only compute the 3D skeleton of a tree, but also obtain its graph structure and radius of the branches. However (and unlike us), they segment the tree using a clean point cloud with a constant background. In general, all these methods work only with the point clouds of isolated trees which were previously pre-processed or cleaned.

One can also find some works that use 2D images to recover the morphology of plants and trees. For example, Zheng et al. [62] segment plants using mean-shift, color features and a shallow neural network to judge whether a cluster belongs to the plant or not. In Zheng et al. [61], they also use mean-shift to segment plants in a field but they employ a Fisher linear discriminant (FLD) instead of a neural network to perform the segmentation. However, none of these methods is evaluated with plants in a real garden. They only focus on green vegetation planted in soil. So they would not be applicable in our case, where it is necessary to deal with a variety of rose bushes, with different branch colors, with or without leaves, and in multiple scenarios. Barth et al. [6] segment the stems of sweet pepper plants along with other 6 classes (buds, leaf stems, cuts, etc.) using a Fully Convolutional Network (FCN) with a Conditional Random Field (CRF) layer. They also generate synthetic images [5] to train the FCN and fine-tune it with a small dataset of real images. The stem segmentation had a low true positive rate because of the similarity among the different classes considered. In addition, the synthetic dataset only includes one environment and focuses on recreating a specific garden. In our case, five different datasets are evaluated, also including a synthetic datasets but with multiple rose bushes viewed from different perspectives, scenarios, and lighting conditions. Gürel et al. [25] and Gürel et al. [24] segment rose stems to find their center lines and trace them with a manipulator using a stereo camera. This process is done using a single rose stem under a controlled environment where the background is different from the stems. Botterill et al. [10] obtain the 3D model of grape vines using 3 monocular cameras. It takes the color information of the cameras, segments them and finds their correspondences to reconstruct the 3D plant. Finally, they move the cameras parallel to the vines to add more information to the 3D reconstruction. The method is robust in constrained environments with constant light conditions.

Another common approach to recover the morphology of a plant is to reconstruct it in 3D using multiple 2D images captured from different perspectives. For example, Isokane et al. [32] segment the plant from 2D views and get the probability of the existence of a branch by using a variation of a Pix2Pix GAN [33], then they reconstruct the plant in 3D combining multiple views. The algorithm works well with synthetic data, however, the method does not generalize well for real data. Another work that combines 2D and 3D images is [42], which captures 64 views of a plant in 2D and obtain the mesh for each view. Unlike the previous methods, they segment the meshes of the plant rather than segmenting the 2D image. For this, they apply a region growing algorithm and fit geometric primitives to the segmented meshes. Simek et al. [51] proposed an approach based on temporal information and Gaussian processes to model the branches from multiple 2D images. Similarly, Gélard et al. [21] build the 3D model of a plant using structure from motion. Then, they segment the plant by fitting a cylinder from the base of the stem until it reaches the top. After segmenting the stem, they remove it from the point cloud and process the leaves by clustering the remaining points. Santos et al. [47] also uses structure from motion to obtain the point cloud from multiple 2D images. They segment each part of the plant by using spectral clustering on the 3D points. Alenya et al. [2] combines color and depth images to segment the leaves of a plant. They perform a rough segmentation of the leaves using a graph-based method [19], then they choose the best leaf segments by fitting quadratic surface models to the segmented depth images and evaluating which segments best fit the depth data. These segments are then post-processed using a nearest-neighbor graph to reduce over segmentation. All these methods extract the point cloud by capturing multiple images of the plant from different perspectives. Also, they operate in the 3D space to separate the branches from the rest of the plant. In our case, we use 2D information combined with a calibrated stereo camera to segment and obtain the 3D morphology of the plant.

In summary, while all these methods segment the plants and obtain their morphology, they usually make strong assumptions. The methods using point clouds usually require the data to be accurate, complete and without noise. This requires much pre-processing work which most of the times is done manually. In the case of works with 2D data, a controlled environment with a fixed background is mainly considered for each perspective of the plant. Among these methods, some of them reconstruct the plant in 3D, however, they have to use multiple views to recover it properly. Moreover, the evaluation of these model based methods usually assumes an error margin which increases their overall performance compared to a result using a pixel error metric.

Another important disadvantage of these methods is the assumption of having a controlled environment to perform the extraction of the plant and do not evaluate using real datasets, with different types of plants and environments. In addition, most of them use images of indoor plants with homogeneous background and/or constant light conditions [10, 24, 25]. On the other hand, the datasets that do contain a real garden usually cover the background of the plant with a constant color [5, 6]. We highlight the importance of using these methods in a real garden because their main applications will require the robot visual system to work in an uncontrolled environment. After exploring the different datasets that previous garden robotics methods use, we found that most of them, apart from the disadvantages mentioned above, are small and lack variability, meaning that they only have a small quantity of images of a single type of plant or tree, making it difficult to evaluate how well those methods generalize compared to other datasets.

Our approach tries to overcome all these disadvantages by using a method without making assumptions about the type of environment or lighting conditions. Moreover, it is validated using five different datasets, with more than 6500 images of indoor and outdoor scenes, including real garden images in different environments, with variable lighting conditions, and without homogeneous backgrounds.

3. Method

The proposed approach to reconstructing a rose bush from stereo images is divided into several steps (see Figure 2): First, the left stereo image is segmented using a Fully Convolutional Segmentation Network (FCSN). In parallel, the stereo pair is supplied to a disparity method to calculate the depth of the plant. These results are post-processed to combine the segmentation and the disparity,

and to calculate the branches that make up the plant. Finally, the 3D reconstruction is obtained using the previous results. These steps will be explained in detail in the following sections.

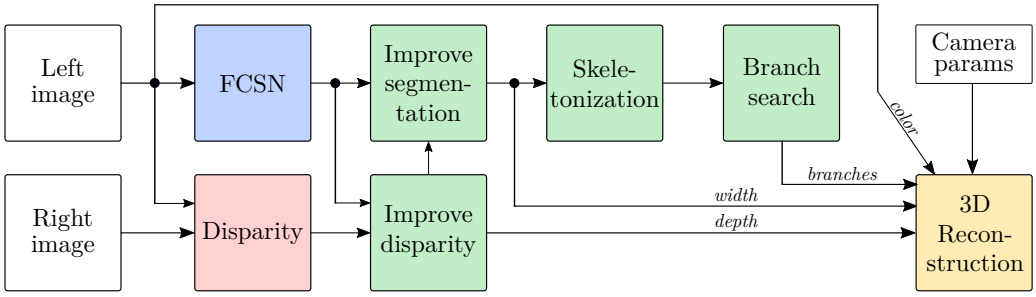


Figure 2: Scheme of the pipeline followed by the proposed method. First, the segmentation and disparity images are calculated. Post-processing is then carried out to combine these results and recover the morphology of the plant. Finally, the 3D reconstruction is calculated using the previous results.

3.1. Fully Convolutional Segmentation Network (FCSN)

The segmentation process is the most important step in the entire proposed algorithm, since the accuracy of the final result will depend on it. Basically, this process performs a classification of the image pixels into two possible categories, indicating if the pixel belongs to the class branch or to the class not-branch, which we denominate as background. In this case, the background not only includes the garden background, but also other parts of the bush itself, such as leaves or flowers, and even other elements that can surround it and that could be similar to a branch, such as sticks of a fence, support stakes, etc. Therefore, this step eliminates everything that is not a branch and thus avoids sending incorrect coordinates to the robot.

It should be considered that branches are very thin objects with colors that can be very similar to the background. In addition, they can also be confused with other surrounding elements (such as a stick or a support stake), especially in this context, in which the images are not captured in a controlled environment with constant background and lighting. All this makes this task considerably more complex than a general segmentation task, in which larger and more prominent objects in the image are usually considered.

To perform the branch segmentation, we use a Fully Convolutional Segmentation Network (FCSN), based on the work of Long et al. [39], but instead of having a categorical vector to indicate the class of each pixel (as other networks of this type also make, such as SegNet [4]), we modify the last layer to return a matrix representing a probability map of the presence of a branch in each pixel of the input image. In other words, the proposed FCSN is trained to perform a function such that $s : \mathbb{R}^{(w \times h)} \rightarrow [0, 1]^{(w \times h)}$, learning a map over a $w \times h$ input image that preserves the input shape and indicates the probability that each pixel belonging to the branch class.

As in the architecture proposed by Long et al. [39], the layer hierarchy of our FCSN follows the idea of auto-encoders, where first a series of convolutional layers combined with pooling layers are added to reduce the size, until an intermediate layer in which a meaningful representation of the input is attained. As these layers are applied, filters are able to relate parts of the image that were initially far apart. This first part of the network would be equivalent to the encoding stage of the auto-encoder (see Figure 3). Then, it follows a series of convolutional plus upsampling layers that reconstruct the image up to the same input size (this second part would be the equivalent of decoding stage). The last layer consists of a set of neurons with sigmoid activation that predict a value in the range of $[0, 1]$, depending on the *selectional* threshold δ for the corresponding input feature. The δ parameter is a hyperparameter that is also learned during the training phase.

In addition, a series of modifications were made to the network architecture and the layers used. The downsampling in the network encoder part is performed by convolutions using stride, instead of resorting to pooling layers. Up-sampling is achieved through transposed convolution layers [59], which perform the inverse operation to a convolution, to increase rather than decrease the resolution

of the output. Residual connections were also added to improve the accuracy of the reconstruction. For this, the encoder layers are connected with the corresponding decoder layers, similar to how it is done in U-Net [46]. But unlike the latter, instead of concatenating the feature maps we add them as is done in other architectures, such as in ResNet [27]. In this way, we can both help the training process and improve the accuracy of the reconstruction.

Feature maps are zero padded so that the dimension before and after the convolution remains the same. Batch normalization [31] is performed after convolution to compensate for the covariance shifts and prevent overfitting during the training procedure. Dropout [53] layers with a probability of 0.5 were also added after each normalization layer to improve the generalization capabilities of the network. Finally, ReLU [22] was used as activation function for all layers except for the output layer, for which the sigmoid activation function is used as explained above.

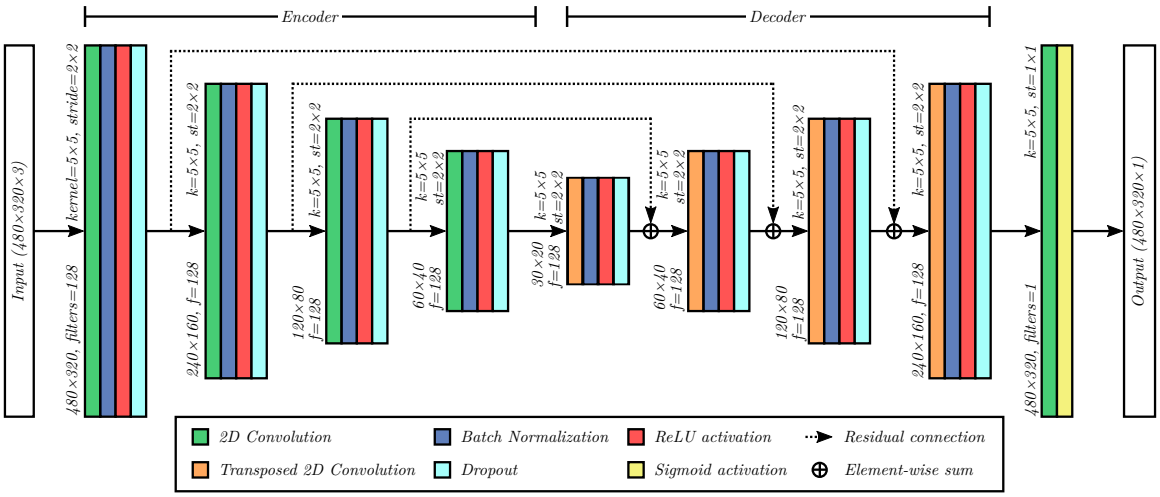


Figure 3: Scheme of the Fully Convolutional Segmentation Network. In this figure, the layer type is labeled with colors according to the legend. The size of each layer for convolutions and transposed convolutions is $h \times w$, where h is the height and w the width. The number of filters (f), the kernel size (k) and the stride value (st) applied for each layer are also shown.

To find the best network configuration for this particular problem, we applied a *grid-search* technique [8], analyzing different values of hyperparameters, including the number of layers of the network, the input size, the number of filters of each convolution, the kernel size, the normalization and the equalization types, the data augmentation factor, the dropout value, and the threshold δ value. The results of the hyperparameters exploration are included in Section 5.1, although we summarize the best topology found for this network in Table 1. Figure 3 also shows an outline of this architecture with the content of each layer.

Table 1: Best hyperparameters found after the grid-search process for the segmentation network FCSN.

Input image size:	480×320 px
Number of encoder/decoder layers:	4+4
Filters per layer:	128
Kernel size:	5×5
Normalization type:	Standard
Equalization type:	HSV
Data augmentation:	10 %
***Selectional* threshold δ:**	0.3

Once the FCSN has been trained, detecting branches from an input image consists of feeding the

image through the FCSN, which outputs the branch probability assigned to each input pixel. Those pixels whose selection value exceeds a certain threshold δ are considered to belong to a branch, whereas the others are discarded.

The training stage consisted of providing the FCSN with examples of images and their corresponding segmentation ground-truth, that is, binary maps over the pixels that belong to branches (see Figure 7c). The binary *cross-entropy* loss function between each output activation and its expected activation was used to calculate the error. The tuning of the network parameters was performed by means of back-propagation using stochastic gradient descent [11] and considering the adaptive learning rate proposed by Zeiler [58]. The training stage lasted a maximum of 300 epochs with a mini-batch size of 8 samples, and *early stopping* when the loss did not decrease during 15 epochs.

In addition, we applied a fine tuning process during the training stage, initializing the network with the weights learned using a synthetic dataset. Data augmentation [36, 13] was also used to artificially increase the size of the training set by randomly applying different types of transformations to the original training samples. This technique usually improves the performance and helps reduce overfitting. In our case, for each image of the training set, 10 augmented images were generated. The transformations applied were randomly selected from the following set of possible transformations: horizontal flips, horizontal and vertical shifts ([-10, 10]% of the image size), zoom ([-10, 10]% of the original image size), and rotations (in the range [-5°, 5°]).

3.2. Disparity calculation

The classic Block Matching (BM) algorithm [35] is proposed to calculate the disparity of the branches. It applies a LOG transform and uses L1 norm correlation to calculate the sum of absolute differences (SAD) using variable disparity search. It also applies a post-filtering with an interest operator and a left/right check.

Although this method is not as accurate as others, like Semi-Global Block Matching (SGBM) [28] or DispNet [40], it gets quite competitive results with a much faster runtime. BM obtains the disparity maps in real-time, whereas SGBM runs at $\sim$ 4 FPS and DispNet at $\sim$ 2 FPS. The fast runtime is necessary to update and keep track of the cutting points and the shape of the bush after each clipping because the branches can be slightly moved by the wind or the manipulator.

The BM method from ROS Kinetic (Robot Operating System) [44] was used to do the stereo matching. Considering that the final prototype of the robot has a stereo camera with a small baseline (0.03m) and that the manipulator's tool-tip is around 0.15m away from the camera [56] (as seen in Figure 1), we are only interested in objects that are equal or farther than that distance. Therefore, the parameters of correlation window size and disparity search windows were set to 15 pixels and 64 pixels respectively.

Once the disparity map is obtained, Equation 1 is used to convert the disparities d (in pixels) into real depth values z (in metres).

$$z = f\frac{B}{d} \tag{1}$$

where f is the focal length of the camera (in pixels) and B the baseline or distance between the two lenses (in metres). In our case, the disparities were obtained using rectified images and the camera parameters. These parameters were found using the calibration software Kalibr [20].

In addition, thanks to the post-processing step that combines the segmentation and the disparity (which will be presented in the next section), we can improve the accuracy of the disparity calculated by this algorithm and obtain a dense disparity for the segmented branches.

3.3. Combine disparity and segmentation

Although the result obtained by the segmentation network is good, sometimes it cannot segment a whole branch completely but it splits it in small regions, as observed in Figure 4(b) inside the red box. These small regions are mostly caused by thin branches or complex areas where it is difficult to separate the background from the foreground. We propose that the segments that belong to the same branch can be joined using the disparity information. This is based on observing the output

of the BM algorithm, which is able to find the disparities of a whole branch, as Figure 4(c) shows. In addition, the BM algorithm creates "blobs" with similar disparities (usually with a size slightly larger than the branch), which allows us to use the disparity image to complete regions where the segmentation is not continuous.

To join the segmented regions, first, each "blob" of the disparity map is evaluated to find if it contains any segmented branch within its boundaries. In the case that a "blob" has two or more segmented branches, these segments are considered part of the same branch and joined if there is at least one linear connection between the pixels of the segments, as seen in Figure 4(d). Using this criteria, only the segmented regions that have similar disparities and are close to one another are joined.

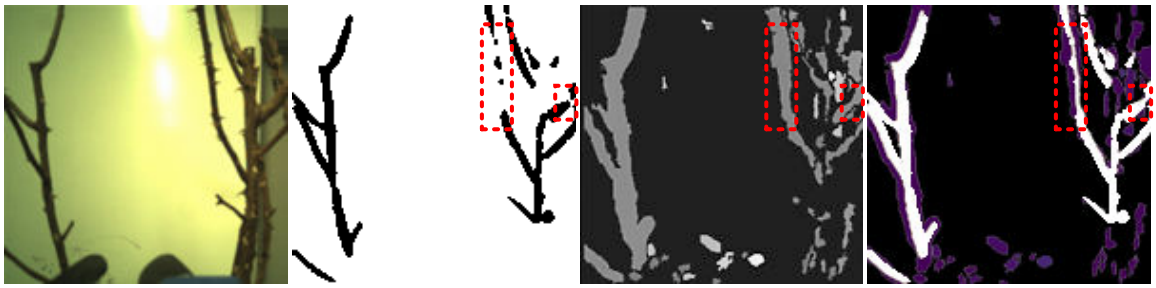


Figure 4: Process of the segmentation completion using the disparity map, where (a) shows the input image, (b) the segmentation obtained by FCSN, (c) the disparity output of the BM algorithm, and (d) how the regions of the branch that were not segmented completely (red rectangles) were joined. To facilitate the visualization of this process, the δ threshold was modified to generate visible errors produced by the segmentation.

On the other hand, classical stereo correspondence methods, like BM, cannot match certain parts of the image and leaves them without any disparity value. This causes some regions, like parts of the branches, to not have depth information. We propose to alleviate this problem by using the segmented image obtained in the previous step as prior information for disparity completion.

The proposed method is the following. First, the coordinates of all pixels that were classified as branches and do not have a disparity value are obtained. The value of these pixels is calculated by interpolating the disparities of their neighbors with an inverse weighted distance. This means that the pixels that are closer to the one that is being evaluated will contribute more than the pixels that are farther away. In the case that the missing pixel does not have a neighbor with disparity, it is discarded. An example of the result obtained by this process can be seen in Figure 13.

3.4. Skeletonization

As next step, the improved segmented image is processed further to get the morphology of the plant. For this, a 2D skeleton is extracted from the binary image.

To find the skeletonization method that suits better to our task, we evaluated five methods: Zhang & Suen [60], Parallel thinning [23], 3D skeletonization [37], Medial axis [9], and RUSTICO [55] (see Section 5.3 for algorithms details and evaluation results). These methods succeed in finding the skeleton of the plant. However, most of them generated small branches that do not represent a real branch but noise or a "small bump" on the edges of the segmented plant. Finally, the method that obtained the best results was Zhang & Suen [60], as seen in Figure 14, both in the accuracy of the calculated skeleton, in the number of small branches generated, and in the runtime. Therefore, we selected this method for the final implementation of the algorithm.

3.5. Branching search algorithm

Once the 2D skeleton is obtained, the branches of the plant are found by exploiting the basic principle of thinning: *A thinning algorithm reduces the components of a binary image to single pixel thickness lines*. This means that, if a pixel belongs to a branch, it should have at most two

neighbors, in case it has more than two neighbors, it should be considered as a branching node, as seen in Figure 5.

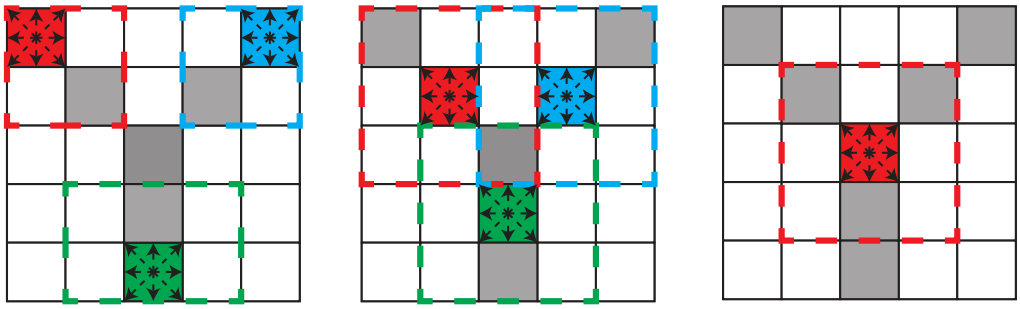


Figure 5: Neighborhood evaluation criteria. In the first and second images, the pixels marked in red, blue, and green are considered as branch pixels because they only have 2 neighbors at most. In the third image, the red pixel is considered as a branching node because it has 3 neighbors.

The proposed algorithm solves the branching as follows. First, it picks a random pixel from the skeleton of the plant. If the pixel has more than 2 neighbors in an 8 neighborhood criteria, it is classified as a node. If the pixel has 2 or less neighbors, it is classified as a branch. To find the rest of the pixels in that branch, it explores the neighbors of the pixel in a recursive way. This means that, if we classify a pixel as a branch, we will evaluate its neighbors until the new neighbors are either classified as a node or they no longer have more neighbors. In addition, to correct possible errors of the skeletonization process, branches with lengths less than or equal to 3 pixels are eliminated.

Algorithm 1 shows the formalization of this process using pseudocode, where the function "*neighbors*" returns the set of neighbors of a pixel, the function "$\mathcal{U}$" returns the set of unvisited pixels, "*find_branch*" is a recursive function that calculates the pixels that make up a branch from an initial pixel, and the sets $\mathcal{N}$ and $\mathcal{B}$ contain the lists of branching nodes and branches found, respectively. In the case of set $\mathcal{B}$, the algorithm creates a sub-list of pixels for each branch.

Algorithm 1: Branching search

$\mathcal{S} \leftarrow$ Rose bush skeleton image
$\mathcal{N} \leftarrow \{\varnothing\}$ ▷ List of branching nodes
$\mathcal{B} \leftarrow \{\varnothing\}$ ▷ List of branch pixels
while $|\mathcal{U}(\mathcal{S})| > 0$ **do**
  $p \leftarrow \mathcal{U}(\mathcal{S})$ ▷ Extract a random unvisited pixel
  $\mathcal{N}, \mathcal{B} \leftarrow find\_branch(p, \mathcal{N}, \mathcal{B})$
end
function *find_branch(p, $\mathcal{N}, \mathcal{B}$)* **is**
  if $|neighbors(p)| > 2$ **then**
    $\mathcal{N} \leftarrow \mathcal{N} \cup \{p\}$
  else
    $\mathcal{B}' \leftarrow \{p\}$
    foreach $p' \in \mathcal{U}(neighbors(p))$ **do**
      $\mathcal{N}, \mathcal{B}' \leftarrow find\_branch(p', \mathcal{N}, \mathcal{B}')$ ▷ *Find rest of the branch*
    end
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathcal{B}'\}$
  end
  return $\mathcal{N}, \mathcal{B}$
end

3.6. 3D reconstruction

Finally, we reconstruct the plant in 3D using the information obtained in the previous steps. Algorithm 2 shows the functional definition of the complete algorithm. As seen, the reconstruction ($\mathcal{R}_{3D}$) is performed using the left input image ($\mathcal{I}_L$), the improved segmentation (S') and disparity ($\mathcal{D}'$) images, the branches ($\mathcal{B}$) found by the Algorithm 1, and the intrinsic parameters of the calibrated stereo camera ($\mathcal{C}_{params}$).

Algorithm 2: Functional definition of the full algorithm

$\mathcal{I}_L, \mathcal{I}_R \leftarrow$ Stereo input images
$\mathcal{S} \leftarrow FCSN(\mathcal{I}_L)$ ▷ Section 3.1
$\mathcal{D} \leftarrow Disparity(\mathcal{I}_L, \mathcal{I}_R)$ ▷ Section 3.2
$\mathcal{S}' \leftarrow Improve\_segmentation(\mathcal{S}, \mathcal{D})$ ▷ Section 3.3
$\mathcal{D}' \leftarrow Improve\_disparity(\mathcal{D}, \mathcal{S}')$ ▷ Section 3.3
$\mathcal{K} \leftarrow Skeletonization(\mathcal{S}')$ ▷ Section 3.4
$\mathcal{B} \leftarrow Branching\_search(\mathcal{K})$ ▷ Section 3.5
$\mathcal{R}_{3D} \leftarrow 3D\_reconstruction(\mathcal{I}_L, \mathcal{S}', \mathcal{D}', \mathcal{B}, \mathcal{C}_{params})$ ▷ Section 3.6

To calculate the 3D reconstruction of the plant, each branch pixel found by the branching search algorithm is represented by a set of features (see Figure 6), these are: its coordinates (x, y, z), diameter or width of the branch (w), and average color of that area of the branch (c).

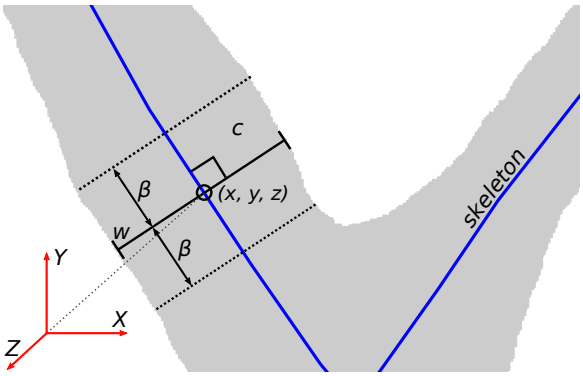


Figure 6: Features extracted for each point of a branch: position (x, y, z), width (w), and principal color (c) in a region of size 2β around the point.

To calculate the color of a branch area, we extract a region of size 2β around a point (x, y) from the original input image, and, within this region, we calculate the dominant color using the color quantization method proposed by Orchard & Bouman [41].

To calculate the width w of the branches, firstly we transform the skeleton into a set of lines. We do this by using the probabilistic implementation of the Hough transform [50], allowing a small gap (3 pixels in our implementation) between pixels to create the lines. We also set the threshold parameter to 10 (minimum number of intersecting points to detect a line) and the minimum segment length to 3 pixels in order to better adjust the lines according to the curvature of the branch. Once the line is obtained, we calculate a perpendicular line to each Hough line. These perpendicular lines start and finish at the boundaries of the segmentation image (see Figure 6).

The conversion from disparity values into a real depth values can be performed directly on the basis of data obtained during the calibration process. Thus, the 3D coordinates (X, Y, Z) of the plant projected in the 3D space are calculated using the row x and column y of pixels in the 2D image, and their corresponding depth values z (obtained from the disparity map using Equation 1).

Equation 2 shows how to calculate this equivalence.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} \frac{z(x-c_x)}{f} \\ \frac{z(y-c_y)}{f} \\ z \end{pmatrix} \tag{2}$$

where c_x and c_y are the principal point (image center) and f is the focal length of the camera (in pixels).

The obtained reconstruction is processed to determine the branches to be cut according to a series of pruning rules. In particular, the information obtained allows us to analyze the rose bush to select the cut points according to different criteria, such as those that exceed a certain height, grow towards the center of the plant, or have a given branch thickness or color (dry branches usually have dark colors).

4. Datasets

For the evaluation of the proposed method we used a total of five datasets, two downloaded from the state of the art (called TB-Roses v2 [54] and Arabidopsis [51]), and three datasets created by the authors, which jointly are called ROSeS (*Roses for Object Segmentation and Skeletonization*)[2]. These three datasets are divided into: S-ROSeS, a synthetic dataset of rose bushes, H-ROSeS, a hybrid dataset with 200 real indoor images but with realistic plastic plants, and R-ROSeS, a completely real dataset with 100 photos of rose bushes taken in different botanic gardens.

H-ROSeS and R-ROSeS were captured using the same stereo camera, with a resolution of 720×480 px, an interocular distance (base line) of 0.03m. The segmented ground truth was obtained by labelling manually each image at the pixel level.

S-ROSeS consists of 5760 stereo images with a resolution of 720×480 px obtained from 160 different views of 36 synthetic rose bushes generated with Blender[3]. The images were generated at different distances and perspectives by rotating the camera around each plant and moving the camera closer and farther from it. Each of these bushes is unique. They were created following the morphology of real rose bushes with a mean height of 0.6m ± 0.20m. The 36 synthetic bushes are distributed in 4 different outdoor environments (9 bushes per environment). The environments differ in background, light conditions and position of the sun. Each stereo pair has its corresponding ground truth with the segmented image, the disparity map, and the skeleton for both left and right views (see Figure 7). These images were also generated using Blender's properties. To obtain the depth map of the synthetic dataset, we simulated a stereo camera with a parallel stereoscopic configuration, a sensor size of 32 mm, a focal length of 0.035m, and a baseline of 0.03m.

TB-Roses v2 dataset [54] is composed of 319 images of rose bushes recorded in a real garden with a resolution of 960×540 pixels[4]. It was designed for testing algorithms for segmentation and delineation of rose branches in applications of gardening robotics. The images are provided together with the ground truth marking the segmented branches.

The Arabidopsis dataset [51] consists of 160 images with a resolution of 2208×1656 px of twelve *Arabidopsis* genus plants taken indoor[5]. The images were taken by rotating a turntable by 10 degrees of yaw. This dataset includes the ground truth with the segmentation and the branching structure.

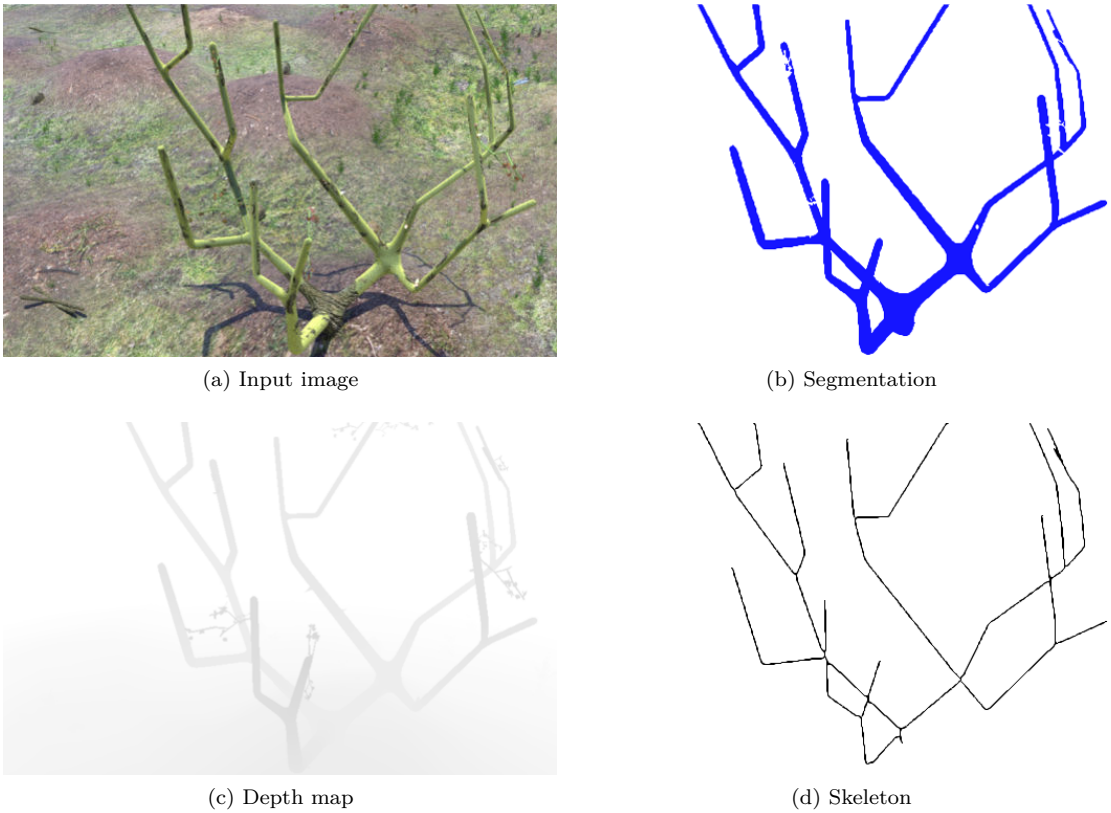
Figure 8 shows some example images of the datasets. For S-ROSeS, four examples with four different backgrounds are included (see Figures 8a, 8b, 8c, and 8d). For the rest, several examples per dataset are also included (Figures 8e to 8l). As can be seen, we collected a variety of datasets, not only in the type and morphology of the plants but also in the type of environment (including

[2]The three datasets of ROSeS are available for the scientific community at `http://trimbot2020.webhosting.rug.nl/resources/public-datasets/`

[3]Blender (`https://www.blender.org/`) is a free and open source 3D creation suite.

[4]TB-Roses v2 is publicly available at: `https://gitlab.com/nicstrisc/RUSTICO/tree/master/data`

[5]Arabidopsis dataset is publicly available at: `http://kobus.ca/research/data/eccv_16_plants/index.html`



(a) Input image (b) Segmentation

(c) Depth map (d) Skeleton

Figure 7: Ground truth of the S-ROSeS dataset, where (a) is the synthetic input image, (b) is the pixelwise segmentation, (c) is the depth map, and (d) is the skeleton of the plant. These example images are from the view of the left camera. The dataset also includes the images obtained from the right camera view.

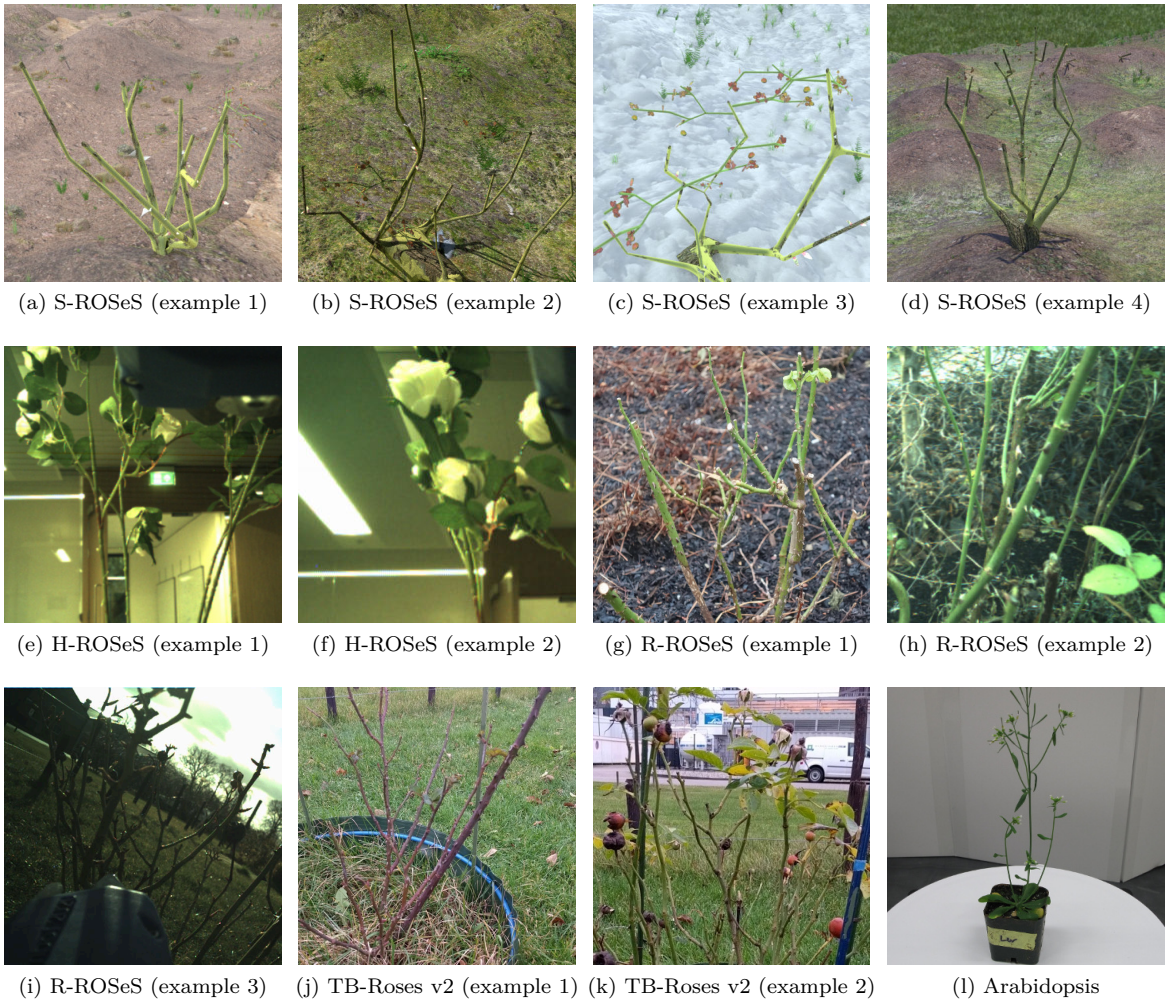
indoor and outdoor), backgrounds (without always using a homogeneous color), and different lighting conditions (Figures 8d and 8h are darker, and the camera in Figure 8i faces the sun).

Table 2 shows a summary of characteristics of the different datasets used, including their type, number of samples, resolution, brightness, and types of ground truth included. As can be seen, some types of ground truth are not available for all datasets. Segmentation ground truth is the only one that is available for all, but the disparity and the skeleton are only included in S-ROSeS. So the validation in each case can only be done for the available data. This table also includes the maximum and minimum average image brightness values per dataset (in the range [0, 255]). These values clearly show how the outdoor datasets have much more variable lighting.

In all the experiments we used an n-fold cross validation, which yields a better Monte-Carlo estimate than when solely performing the tests with a single random partition [34]. Therefore, the datasets were divided into n subsets, using, for each fold, one of the partitions for test (with $1/n$ of the samples) and the rest for training $(1 - 1/n)$.

Partitions were created by separating the datasets according to the sequences or backgrounds used, with the intention of creating mutually exclusive subsets. For example, for S-ROSeS, $n = 4$ partitions were created (1 for each type of background), for H-ROSeS $n = 2$ partitions (corresponding to the two different sequences), for R-ROSeS $n = 4$ (since the images were taken in two different gardens and for each one 2 sequences were recorded), for TB-Roses $n = 3$ (corresponding to three sequences of images), and for Arabidopsis $n = 4$ (with 3 plants per partition, since it has 12 plants in total).

For tuning the hyperparameters (see Section 5.1), the training partition was divided into two, assigning 10% of these samples for validation and the rest for training. The classifier was trained and evaluated n times using these sets, after which the average results plus the standard deviation



(a) S-ROSeS (example 1) (b) S-ROSeS (example 2) (c) S-ROSeS (example 3) (d) S-ROSeS (example 4)

(e) H-ROSeS (example 1) (f) H-ROSeS (example 2) (g) R-ROSeS (example 1) (h) R-ROSeS (example 2)

(i) R-ROSeS (example 3) (j) TB-Roses v2 (example 1) (k) TB-Roses v2 (example 2) (l) Arabidopsis

Figure 8: Example images of the different datasets used for the evaluation.

σ were reported.

5. Experiments

In this section we evaluate the different parts of the proposed method using the datasets described in Section 4. First, the proposed FCSN segmentation network is evaluated, analyzing its different hyperparameters and comparing it with other state-of-the-art methods (see Section 5.1). Next, the calculation of the disparity and the post-processing step used to combine the segmentation and disparity images are assessed in Section 5.2. Section 5.3 compares five different skeletonization methods used for the detection of branches. Finally, the accuracy of the 3D reconstruction obtained is evaluated in Section 5.4.

All these experiments were performed using a Razer Blade 14 with Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz (4th Gen) with 16 GB DDR4 RAM, a Nvidia GeForce GTX 1070 GPU, and ROS Kinetic with Ubuntu 16.04 as operating system.

5.1. FCSN evaluation

In this section we evaluate the FCSN proposed to perform the segmentation of the branches. First, different network hyperparameters are analyzed. For these initial experiments we used the 4

Table 2: Summary of characteristics of the datasets evaluated, including the number of images, their resolution, their type (real, hybrid or synthetic), the minimum/maximum average image brightness values (in the range [0, 255]), the ground truths (segmentation, disparity and/or skeleton), whether they are indoor or outdoor, and if they have variable lighting.

Name	# Images	Resolution (pixels)	Type	Min/Max average brightness	Segment.	Disparity	Skeleton	Indoor	Variable lightning
S-ROSeS	5760	720×480	Synthetic	87 / 193	✓	✓	✓	–	✓
H-ROSeS	200	720×480	Hybrid	115 / 136	✓	–	–	✓	–
R-ROSeS	100	720×480	Real	70 / 201	✓	–	–	–	✓
TB-Roses v2	319	960×540	Real	85 / 169	✓	–	–	–	✓
Arabidopsis	160	2208×1656	Real	111 / 143	✓	–	–	✓	–

sets of plants from the S-ROSeS dataset. Subsequently, using the best model found, we evaluate the rest of the datasets and compare the proposed method with other approaches from the state of the art.

In order to assess the performance of the proposed method, three evaluation metrics widely used for this kind of tasks were chosen, they are: Precision, Recall, and F_1, which can be defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4}$$

$$F_1 = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FN} + \text{FP}} \tag{5}$$

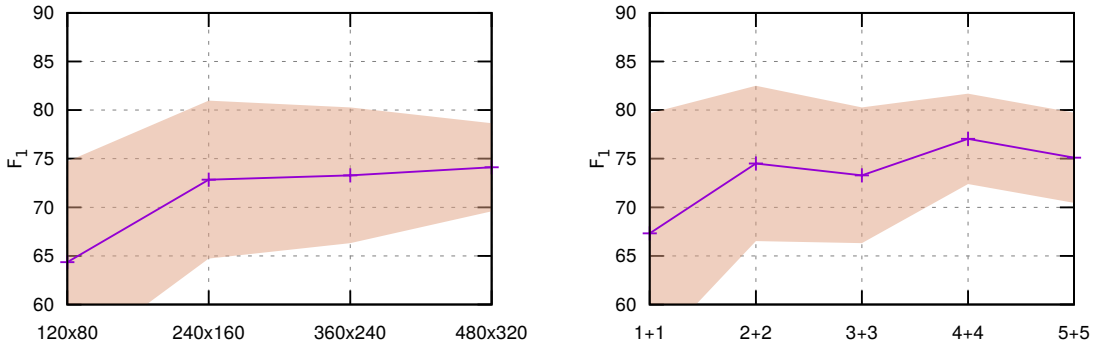
where TP (True Positives) denotes the number of correctly segmented branch pixels, FN (False Negatives) the number of non-segmented branch pixels, and FP (False Positives) the number of background pixels incorrectly given as branch pixels.

5.1.1. Hyperparameters evaluation

To select the best hyperparameters and configuration for the FCSN, we performed a *grid-search* process [8] using the S-ROSeS dataset. The configurations evaluated include variations in the network input size (from 120px to 480px of width and maintaining the original aspect ratio to calculate the height), in the number of layers (from 2 to 10), in the number of filters per layer (between 8 and 128), and in the kernel size (between 3 and 7). In each experiment, only one parameter was changed, setting the rest to a standard network configuration with an input size of 360×240px, 6 layers of depth (3 encoding + 3 decoding layers), 32 filters per layer with a kernel size of 3×3, with a *selectional* threshold δ of 0.5, without equalizing the input image, and only normalizing the input values dividing by 255.

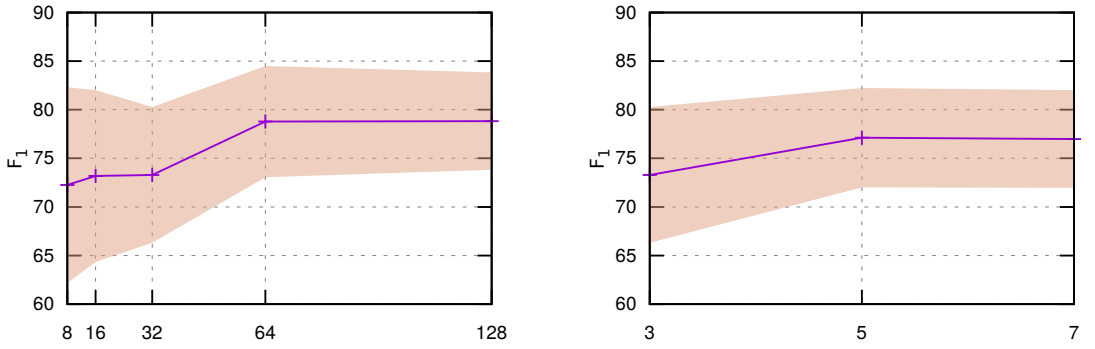
Figure 9 shows the average results plus the standard deviation of these experiments. The reported results are the average of the 4-fold cross validation, where, for each fold, we used one of the S-ROSeS dataset partitions for test (25% of the samples) and the rest for training (75%), without mixing neither the types of plants nor the backgrounds, consequently dividing it into 4 mutually exclusive sub-sets. As stop criterion for tuning the hyperparameters, the training partition was divided into two, assigning 10% of the samples for validation and the rest for training.

The average F_1 when varying the input size is shown in Figure 9a. As seen, the best results (with the lowest standard deviation) were obtained using the larger input size (480×320px). Even larger sizes were also evaluated, but they increased the requirements of the machine, which did not allow their use due to the hardware specifications of the robotic platform. In the case of the number of layers (Figure 9b), an increasing trend is also observed, obtaining the best result with 4+4 layers and then getting worse slightly. The standard deviation also decreases up to 4+4 layers and it then



(a) Input size

(b) # layers



(c) # filters

(d) Kernel size

Figure 9: Average F_1 (%) plus standard deviation (in light red) of the grid-search process when varying (a) the input image size, (b) the number of encoding + decoding layers, (c) the number of filters per layer, and (d) the kernel size of the convolutional filters.

stabilizes. In this case, it was not possible to add more layers due to the base input size (360×240px), since in each layer the size is divided by 2. Figure 9c shows the result obtained when varying the number of filters per layer. In this case, using more filters also increases the F_1 and reduces the standard deviation. This improvement is more noticeable from 16 to 64 filters, increasing more slightly later when adding more filters. Figure 9d shows the average F_1 for the three kernel sizes evaluated, obtaining the best results with a kernel size of 5×5. In this case, the standard deviation hardly varies.

We now proceed to analyze the influence of the normalization type as well as the equalization applied to the input data, setting for this the best configuration found in the previous experiments.

Literature cites different ways to normalize the data used to feed a network [49, 36], but the most appropriate technique depends on the particular problem. The most common normalization methods are:

$$Z_{standard} = \frac{M - mean(M)}{std(M)} \qquad Z_{mean} = M - mean(M)$$

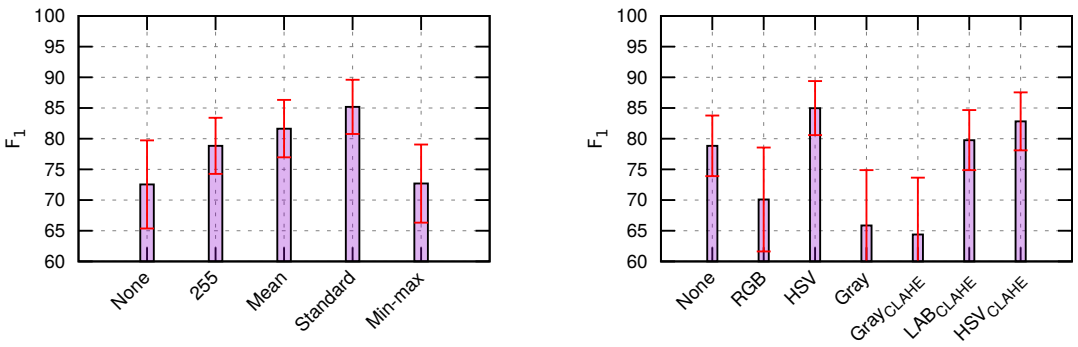
$$Z_{min-max} = \frac{M - min(M)}{max(M) - min(M)} \qquad Z_{norm} = M/255$$

where M is the input matrix containing the raw image pixels from the training set. For the normalization of the test set we used the same mean, deviation, max, and min values calculated for the training set.

Moreover, since it was observed that the image contrast and brightness affected the result ob-

tained significantly, different types of equalization were also analyzed. The equalization of the histogram applies a transformation on the image in order to obtain a histogram with a uniform distribution of colors (or levels of gray) that improves the contrast of the image. For this, the histogram equalization was evaluated using gray values and both RGB and HSV color spaces. In addition, the CLAHE (Contrast Limited Adaptive Histogram Equalization) method [1] was evaluated in gray and both CIELAB and HSV color spaces. This method performs an adaptive equalization by regions controlling the limit of the equalization made to not overamplify noise in homogeneous regions.

We evaluated these types of normalization and equalization on the proposed network, including the option of not normalizing or equalizing the data. Figure 10 shows the average results plus the standard deviation of these experiments. As before, each result is the average of the 4 folds using the best configuration found in the previous experiments and only varying the type of normalization or equalization. The type of data normalization (see Figure 10a) considerably affects the result obtained, since the difference between the best and the worst result exceeds 12%. The best F_1 is obtained using the standard normalization, followed by the mean norm. In addition, in these two cases, the standard deviation is quite similar. For the equalization type (see Figure 10b), a significant difference in the results obtained is also shown. Both gray scale and RGB equalizations seem to worsen the result, however, HSV and CLAHE (both in RGB and HSV color spaces) equalizations does improve it. The best result is obtained with the HSV equalization, which also reports the lowest standard deviation and is better than the option of not equalizing by 6.15 %.



(a) Normalization type

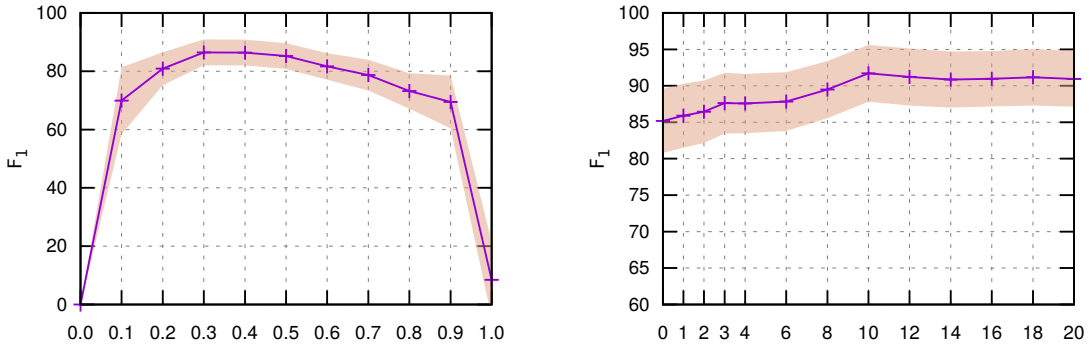
(b) Equalization type

Figure 10: Average F_1 (%) plus standard deviation obtained for the different types of (a) normalization and (b) equalization considered.

Subsequently, we also evaluated the *selectional* threshold δ that is applied to the output of the network to determine (select) the pixels belonging to a branch. Figure 11a shows the result of varying this threshold between 0 and 1. The network obtains results higher than 80% with thresholds between 0.2 and 0.6, obtaining the best F_1 and the lowest standard deviation with a threshold of 0.3.

Finally, the influence of the data augmentation process was evaluated. For this, the number of samples of the training set was artificially increased by applying different types of random transformations to these images (This process is described in Section 3.1). In order to evaluate the improvement obtained with this augmentation process, we carried out an experiment in which we gradually increased the number of random transformations applied to each image from our training set, and evaluated it using the test set. Figure 11b shows the average results plus the standard deviation of such experiment, where the horizontal axis represents the augmentation factor and the vertical axis the F_1 obtained. As seen, the highest improvement is obtained at the beginning, after which the results begin to stabilize and stop improving after 10 augmentations. In this case, the standard deviation is quite stable, it is only slightly reduced by adding the first augmented images.

The finally selected configuration for the FCSN network was an input size of 480×320px, 4+4 layers, 128 filters per layer with a kernel size of 5×5, standard normalization, HSV equalization, a



(a) *Selectional* threshold δ (b) Data augmentation factor

Figure 11: (a) Influence of the *selectional* threshold δ (horizontal axis) on the final F_1 result obtained (vertical axis). (b) Average results of the data augmentation process. The horizontal axis represents the number of augmentations and the vertical axis the average F_1 (in percentage). Standard deviation is also shown in light red.

selectional threshold δ of 0.3, and augmenting the data by a factor of 10 (i.e. generating 10 random transformations for each training image). Table 1 shows a summary with these settings.

5.1.2. FCSN results

Once the best architecture for the FCSN network was obtained, it was evaluated with all the datasets described in Section 4. In addition, we compared it with the results obtained by other state-of-the-art methods, these are: U-Net [46], SegNet [4], and DeepLabv3 [14] (see Appendix A for a description of these methods).

In addition to Precision, Recall and F_1 metrics, the Intersection over Union (IoU) [15] metric was used for evaluation. The metrics previously used are suitable for unbalanced datasets but in some cases they are not the most fair because they measure the precision at the pixel level, but not whether the algorithm has detected a branch or not. So it is possible that thick branches are detected very well but thin branches not, and still obtain good results. It is also possible that the algorithm makes mistakes in the branches' edges and gets a low result with these metrics when, in fact, all the branches are being detected correctly.

The IoU metric helps to measure whether the algorithm correctly detects all the branches and also how well it detects their size and location. To calculate this metric, we map each branch[6] of the ground truth (gt) onto the segmentation proposals (bp) with which it has a maximum IoU overlap according to the following equation:

$$IoU = \frac{\text{area}(B_{bp} \cap B_{gt})}{\text{area}(B_{bp} \cup B_{gt})} \tag{6}$$

where $\text{area}(B_{bp} \cap B_{gt})$ depicts the intersection between a branch proposal and the ground truth, and $\text{area}(B_{bp} \cup B_{gt})$ depicts its union.

We also calculate the metric F_1 at the branch level considering as TP when the IoU value exceeds a certain threshold λ (by convention $\lambda = 0.5$). We consider as FP the wrong detections (i.e., when a B_{bp} does not overlap with any B_{gt}), and as FN when a ground truth branch is not detected. Note that if multiple detections of the same branch are predicted, only the first one is counted as positive and the rest as negatives.

Table 3 shows the results obtained by each of the methods compared. The results have been grouped by dataset, also showing the average at the end of the table. The best result obtained by dataset and on average is marked in bold for each metric.

[6] To perform this process, we manually separated each branch from the ground truth.

As can be seen, the proposed method obtains better results than the rest of approaches, except for the precision with S-ROSeS. However, this result is only 0.6 worse than that obtained by U-Net, and, given the standard deviation values (3.2 and 4.5, respectively), this difference can be neglected. Moreover, if we analyze the F_1 metric at pixel level, it shows that FSCN obtains a better overall segmentation. Therefore, it correctly recovers a greater number of branches, as shown by the metric F_1 at branch level. On average, the proposed method obtains a F_1 8.18 % better than the next best result at pixel level, and 7.33 % if we analyze it at branch level. The datasets for which a greater improvement is obtained with respect to the other methods are Arabidopsis (10.75 % better than the next best result at pixel level), H-ROSeS (8.55 % better), and R-ROSeS (5.98 % better), which are the datasets that present a greater difficulty since they have thinner branches, more complex backgrounds, and a greater number of branches to segment.

Table 3: Results obtained by the different algorithms evaluated for each of the datasets. The best result obtained by dataset and on average is marked in bold for each metric.

		Pixel level			**Branch level**	
Dataset	**Method**	**Precision**	**Recall**	$\boldsymbol{F_1}$	**Avg. IoU**	$\boldsymbol{F_1}$
S-ROSeS	U-Net	**91.08** ± 3.2	88.50 ± 4.5	89.77 ± 3.9	90.21 ± 9.5	93.63 ± 7.6
	SegNet	88.84 ± 2.9	77.07 ± 11.1	81.81 ± 8.3	83.60 ± 7.9	87.79 ± 5.0
	DeepLabv3	76.94 ± 6.2	84.63 ± 5.5	80.59 ± 5.8	79.64 ± 6.8	81.21 ± 6.5
	FCSN	90.47 ± 4.5	**93.18** ± 3.0	**91.70** ± 3.9	**94.16** ± 7.3	**97.09** ± 4.2
H-ROSeS	U-Net	48.27 ± 0.7	66.15 ± 4.3	55.80 ± 2.0	66.11 ± 9.5	76.61 ± 7.1
	SegNet	56.92 ± 1.7	58.95 ± 1.5	57.48 ± 1.6	55.11 ± 11.5	64.98 ± 11.2
	DeepLabv3	50.79 ± 0.3	55.67 ± 0.7	53.12 ± 0.5	61.75 ± 10.7	67.34 ± 11.8
	FCSN	**61.20** ± 1.2	**71.79** ± 3.5	**66.03** ± 2.6	**69.40** ± 8.3	**80.35** ± 7.3
R-ROSeS	U-Net	69.45 ± 8.0	71.00 ± 8.9	70.19 ± 8.4	68.15 ± 13.0	73.88 ± 13.5
	SegNet	70.58 ± 9.5	73.69 ± 8.2	72.08 ± 8.8	70.08 ± 12.5	75.69 ± 12.2
	DeepLabv3	61.09 ± 2.5	61.49 ± 2.9	61.21 ± 2.7	71.85 ± 8.4	74.19 ± 8.9
	FCSN	**77.48** ± 7.8	**78.72** ± 5.6	**78.06** ± 6.6	**76.71** ± 8.7	**82.18** ± 8.2
TB-Roses	U-Net	65.53 ± 4.8	73.74 ± 3.0	69.38 ± 4.1	77.04 ± 6.5	80.09 ± 6.0
	SegNet	69.73 ± 0.4	79.48 ± 2.9	74.27 ± 1.0	82.71 ± 11.9	86.28 ± 12.2
	DeepLabv3	47.45 ± 0.9	59.23 ± 1.0	52.69 ± 0.9	80.89 ± 8.9	82.50 ± 9.0
	FCSN	**75.11** ± 0.7	**80.17** ± 2.4	**77.55** ± 1.5	**84.12** ± 7.0	**87.44** ± 7.7
Arabidop.	U-Net	47.45 ± 0.7	59.23 ± 0.8	52.69 ± 0.8	73.81 ± 10.8	78.06 ± 11.9
	SegNet	54.45 ± 2.9	65.46 ± 7.3	58.52 ± 8.1	77.49 ± 15.8	82.50 ± 16.4
	DeepLabv3	60.91 ± 3.4	61.02 ± 3.4	60.96 ± 3.4	79.14 ± 7.7	85.56 ± 8.2
	FCSN	**71.52** ± 3.1	**72.44** ± 4.2	**71.71** ± 3.5	**87.67** ± 7.1	**91.88** ± 7.3
Average	U-Net	64.35 ± 5.2	71.72 ± 5.1	67.56 ± 5.0	75.06 ± 10.1	80.46 ± 9.7
	SegNet	68.10 ± 5.5	70.93 ± 10.5	68.83 ± 6.5	73.80 ± 12.2	79.45 ± 12.0
	DeepLabv3	59.44 ± 3.4	64.41 ± 3.2	61.71 ± 3.1	74.65 ± 8.6	78.16 ± 9.1
	FCSN	**75.16** ± 5.4	**79.26** ± 5.6	**77.01** ± 4.6	**82.41** ± 7.7	**87.79** ± 7.1

To rigorously analyze the results obtained, we performed a statistical significance comparison by considering the paired sample non-parametric Wilcoxon signed-rank test [17]. More precisely, the idea is to assess whether the improvement observed in the segmentation performance (F_1 score at the pixel level) with the use of the FCSN is statistically significant in comparison with the result obtained by the other methods. For this, the results obtained for each dataset and fold were pairwise compared. By making this comparison, the proposed method obtains a p-value of 0.005, if we compare it with SegNet, and 0.002 with respect to U-Net and DeepLabv3. Therefore, this test reflects that our proposal significantly outperforms (considering a statistical significance threshold of $p < 0.01$, the most restrictive threshold normally used) the results obtained by the other state-of-the-art methods.

Figure 12 shows an example of the segmentation result obtained by each method for a sample image of each of the evaluated datasets. To make this figure more informative, we selected samples in which all methods made visible mistakes. The input image is shown in the first column. The rest of

the columns correspond to the results obtained by each method, where black and white areas depict correct detections of branches and background, respectively, and and red and blue pixels depict FP and FN of branches, respectively. These figures help to visualize the accuracy of the methods and to understand where the errors occur. As can be seen, U-Net and SegNet generate more noise due to non-branch elements (general background, leaves, etc.), while DeepLabv3 tends to make a thicker segmentation. The proposed method also makes mistakes, but they are mainly produced in the contours of the branches.

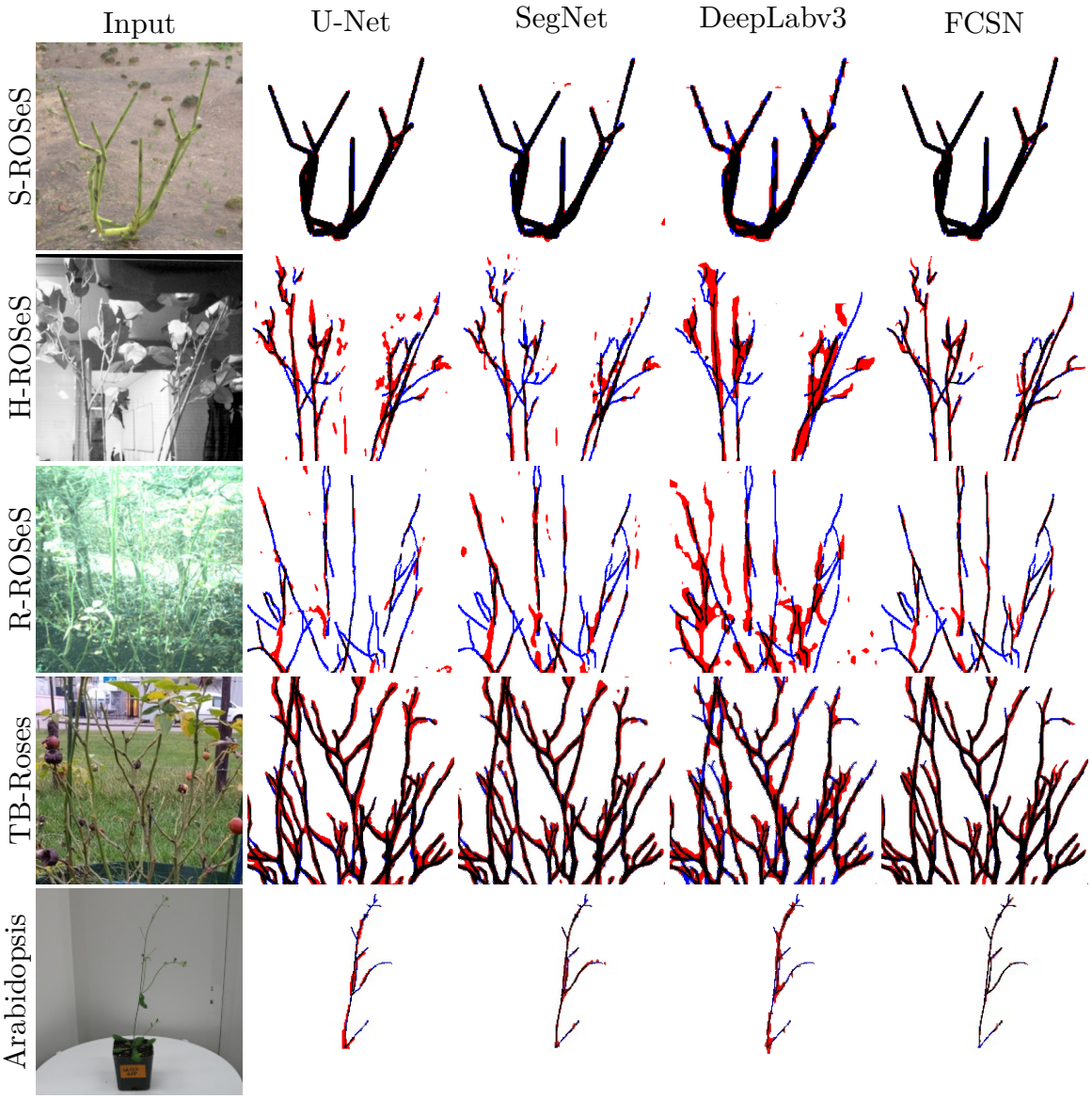


Figure 12: Examples of the segmentation result obtained by each method for a sample image of each of the evaluated datasets. The first column shows the input image. The rest of the columns correspond to the results obtained by each method, where black and white areas depict correct detections of branches and background, respectively, and red and blue pixels depict FP and FN of branches, respectively.

5.2. Assess segmentation and disparity combination process

The post-processing step used to combine the segmentation and disparity images is evaluated in this section. Several depth metrics from prior works [18] are used for the evaluation of the disparity

results, which are:

- Root Mean Squared Error (RMSE): $\sqrt{\frac{1}{T}\sum_i^T ||d_i - d_i^{gt}||^2}$
- Squared Relative difference (Sq-Rel): $\frac{1}{T}\sum_i^T \frac{||d_i - d_i^{gt}||^2}{d_i^{gt}}$

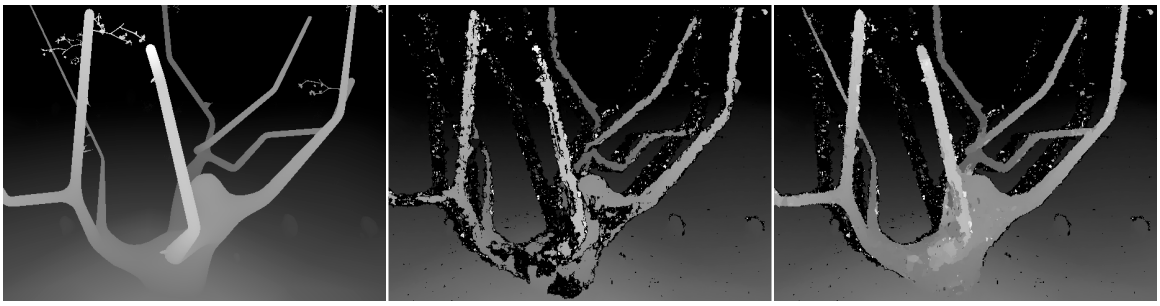
where T is the total pixels in the image, d_i is the estimated depth for the pixel i, and d_i^{gt} is the ground-truth depth.

Table 4 shows the results of this experiment, where the disparity evaluation is shown in the first four columns and the results obtained for segmentation in the last three. The disparity was evaluated for the complete image (first two columns) and also at the branch level (i.e. taking into account only the pixels within the segmentation ground truth, see third and fourth columns). In our case, the results at the branch level are the most interesting, since on the one hand the disparity improvement process only affects branches, and on the other hand, the rest of the process only uses these depth values. As seen, these processes help to improve the original result obtained for both segmentation and disparity (with the BM method). The improvement obtained for the disparity is the most significant, reducing the RMSE from 0.2527 to 0.0869 at branch level.

Table 4: Results obtained for the calculation of the disparity and segmentation with the S-ROSeS dataset before and after applying the segmentation and disparity combination process. For the disparity, the error calculated for the complete image and at the branch level is shown.

	Disparity						
	Full image		**Branch level**		**Segmentation**		
	RMSE	**Sq-Rel**	**RMSE**	**Sq-Rel**	**Precision**	**Recall**	$\boldsymbol{F_1}$
Original	0.5648±0.2	0.0917±0.1	0.2527±0.2	0.0310±0.1	90.47±4.5	93.18±3.0	91.70±3.9
Improved	0.5082±0.2	0.0630±0.0	0.0869±0.1	0.0039±0.0	92.86±4.4	92.59±3.1	92.96±3.8

Figure 13 shows an example of the result obtained after applying the segmentation and disparity combination process. As can be seen in the figure on the right, the final result substantially improves the original disparity (central image, obtained with the BM method), since this method fills the empty areas, obtaining a dense disparity for the bush branches. On this result, the intersection with the segmentation image is then calculated to eliminate all the background noise and leave only disparity values for branches.



GT | Original disparity | Improved disparity

Figure 13: Example of the result obtained through the post-process to improve the disparity. The first image shows the disparity ground truth, the central image shows the original result obtained by the BM method, and the right image shows the the improved disparity obtained after the segmentation and disparity combination process.

5.3. Skeleton evaluation

In this section we evaluate the skeletonization process individually, without considering the errors made in the previous steps. For this, we analyze the result using the segmentation ground truth

as input of this algorithm, leaving the evaluation of the whole workflow for the 3D reconstruction section.

To evaluate the skeletonization, the F_1 score is used by comparing the obtained 3D skeleton S_o and the ground truth skeleton S_{gt} at the pixel level. Because the skeleton is only one pixel wide, a certain tolerance distance d is considered to determine if a pixel is a TP (similar to Zou et al. [63]). Therefore, a pixel is considered to be TP if it is located no more than d pixels away from the ground truth. These experiments were performed using the hardware specifications described at the beginning of this section.

The following five methods were evaluated using this criteria to find the skeletonization method that suits better to our problem: Zhang & Suen [60], Parallel thinning [23], 3D skeletonization [37], Medial axis [9], and RUSTICO [55] (see Appendix B for a description of these methods).

Given that RUSTICO is not defined as a skeletonization method per se and uses as input a gray scale image instead of a binary one, it delineates curved objects from the background as well, as seen in Figure 14h. Because of this, two experiments using this method were performed. One by comparing the raw output of RUSTICO with the ground truth skeleton and other by only considering the skeleton obtained by RUSTICO inside the ground truth segmented image (denoted by RUSTICO+seg), to make a more fair comparison with the other methods (which directly use the segmented image).

Table 5 shows the quantitative results of this experiment using the S-ROSeS dataset and different d values ($d = [0, 1, 2]$). As seen, all the skeletonization methods achieved scores above 0.3, 0.7 and 0.9 for $d = [0, 1, 2]$ respectively, with the exception of RUSTICO and RUSTICO+seg, which got the lowest results. However, these results are similar to those obtained in the original paper for the branch segmentation task ($F_1 = 42.65$ for the TB-Roses v2 dataset with a tolerance of 2 pixels [55]). RUSTICO's low performance is caused by their nature of finding curvilinear structures rather than finding a skeleton of a blob.

On average, the best F_1 results are obtained by the 3D skeleton method and medial axis. However, Zhang & Suen [60] and parallel thinning are only $\sim 1\%$ worse. The main problem with the 3D skeleton is that it is computationally more expensive than the other methods (4.6121 sec. vs. 0.0390 sec. for Zhang & Suen [60][7]) because it has to evaluate regions of $[3 \times 3 \times 3]$ per disparity layer. The problem with the medial axis method is that it gets affected by the noise and bumps at the edges of the segmentation, which creates small branches that connects the main axis with those small protuberances. The parallel thinning method has a similar problem. The method that creates less small branches in the skeleton and runs faster is by Zhang & Suen [60]. Given that we want to implement a method that runs fast and also that creates as few FP branches as possible, we eventually selected this method.

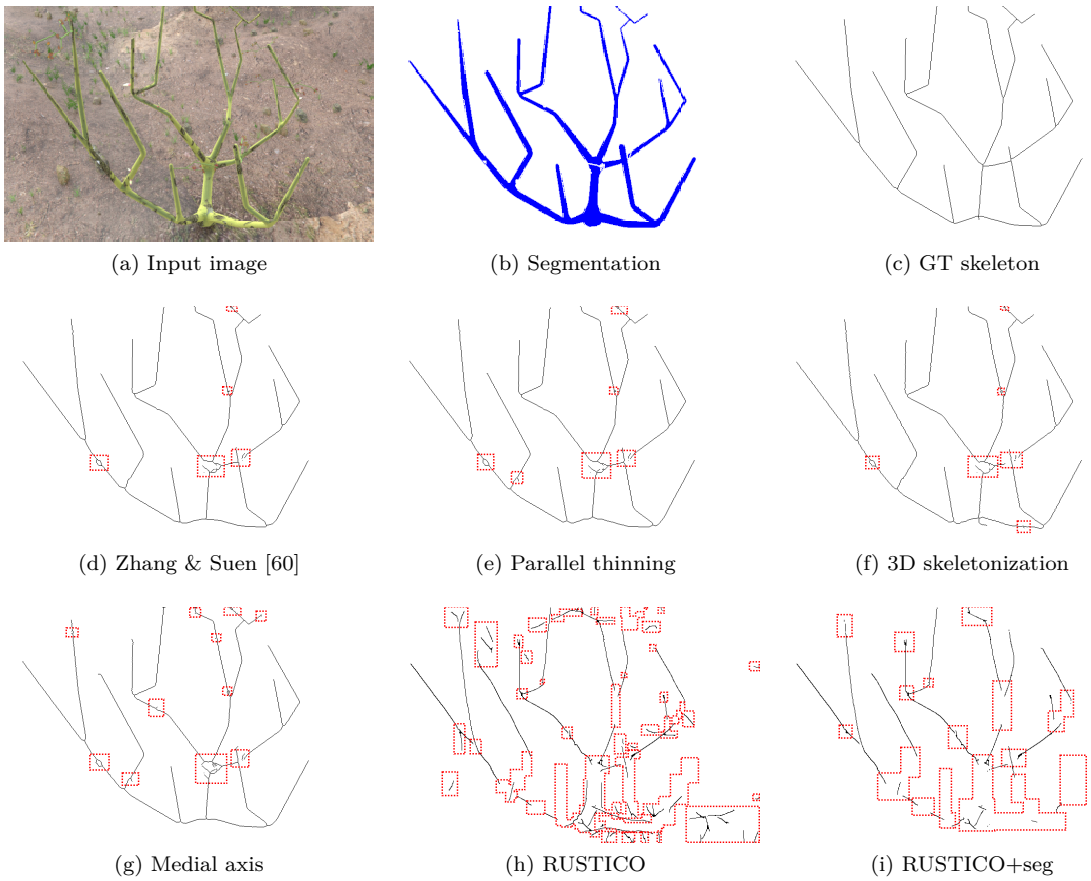
Table 5: Comparison of the results obtained by the different skeletonization methods considered using the segmentation ground truth from the S-ROSeS dataset. The best result obtained for each metric and threshold d is marked in bold.

Method	**Precision**			**Recall**			**F_1**		
	d=0	d=1	d=2	d=0	d=1	d=2	d=0	d=1	d=2
Zhang & Suen	**90.25**	**98.61**	**99.47**	18.47	61.88	83.96	30.66	76.04	91.06
Parallel thin.	90.22	98.60	99.46	18.26	62.06	84.33	30.37	76.17	91.27
3D skel.	87.97	97.66	98.82	**19.05**	**64.99**	**87.40**	**31.32**	78.04	**92.76**
Medial axis	89.68	98.49	99.39	18.92	64.63	86.12	31.25	**78.05**	92.28
RUSTICO	21.74	45.47	54.99	15.08	42.82	60.96	17.81	44.11	57.83
RUSTICO+seg	73.25	93.63	96.81	14.92	42.46	60.35	24.79	58.43	74.35

Figure 14 shows an example of the skeleton obtained by each method for a sample image from the S-ROSeS dataset. Red rectangles mark the errors made by each method. As can be seen, visually (if we compare it with the ground truth shown in Figure 14c) all methods, except RUSTICO, get a good result. RUSTICO correctly detects the main branches but introduces a lot of noise and fails for the small branches (even after intersecting it with the segmentation, RUSTICO+seg). The other

[7] These experiments were performed using the hardware specifications described at the beginning of this section.

four methods work better, obtaining similar results, however, Zhang & Suen [60] generates fewer incorrect small branches.



(a) Input image (b) Segmentation (c) GT skeleton

(d) Zhang & Suen [60] (e) Parallel thinning (f) 3D skeletonization

(g) Medial axis (h) RUSTICO (i) RUSTICO+seg

Figure 14: Examples of the skeleton obtained by each method for a sample image from the S-ROSeS dataset. First row shows: (a) the input image, (b) the segmentation GT, and (c) the skeletonization GT. The other two rows show the skeletonization results. Red rectangles indicate the mistakes made by each method.

5.4. 3D reconstruction

To evaluate the performance of the 3D reconstruction, we compared it with the 3D skeleton ground truth of the synthetic images. This ground truth has 2880 pointclouds, each one containing the 3D skeleton of a synthetic rose. The evaluation consisted of finding the mean of the minimum distance between each point of the ground truth and the reconstructed skeleton. Therefore, this metric measures how far the reconstructed skeleton is, in average, with respect to the ground truth. To make a fair evaluation, the depths of the reconstructed points were normalized by the furthest point in the ground truth per each plant.

Table 6 shows the result of this evaluation. It also shows the distance in each axis x, y and z, where z represents the depth, and y the axis that points to the ground. As can be seen, the average distance is less than 1 cm, being the error in the x-y plane less than the one made in the z-axis (due to the calculation of depth). However, this error is still less than 1 cm. This accuracy is enough for the pruning process because the end effector has an opening size of 1.4 cm and is curved at the tool-tip (as seen in Figure 1), which allows the branch to slide into the center of the cutter even if the branch is not completely aligned with the center of the tool.

Figure 15 shows some examples of the 3D reconstruction obtained with different input images. This figure allows you to visually compare the result obtained by the proposed method and the

ground truth. In addition, the branch separation obtained by Algorithm 1 is included in the 4th column.

Table 6: Average distance between the 3D reconstruction and the ground truth pointclouds.

Min. distance	**Mean (m)**	**std (m)**
x axis	0.00293	0.00477
y axis	0.00336	0.00497
z axis	0.00619	0.00827
Overall	0.00859	0.01001

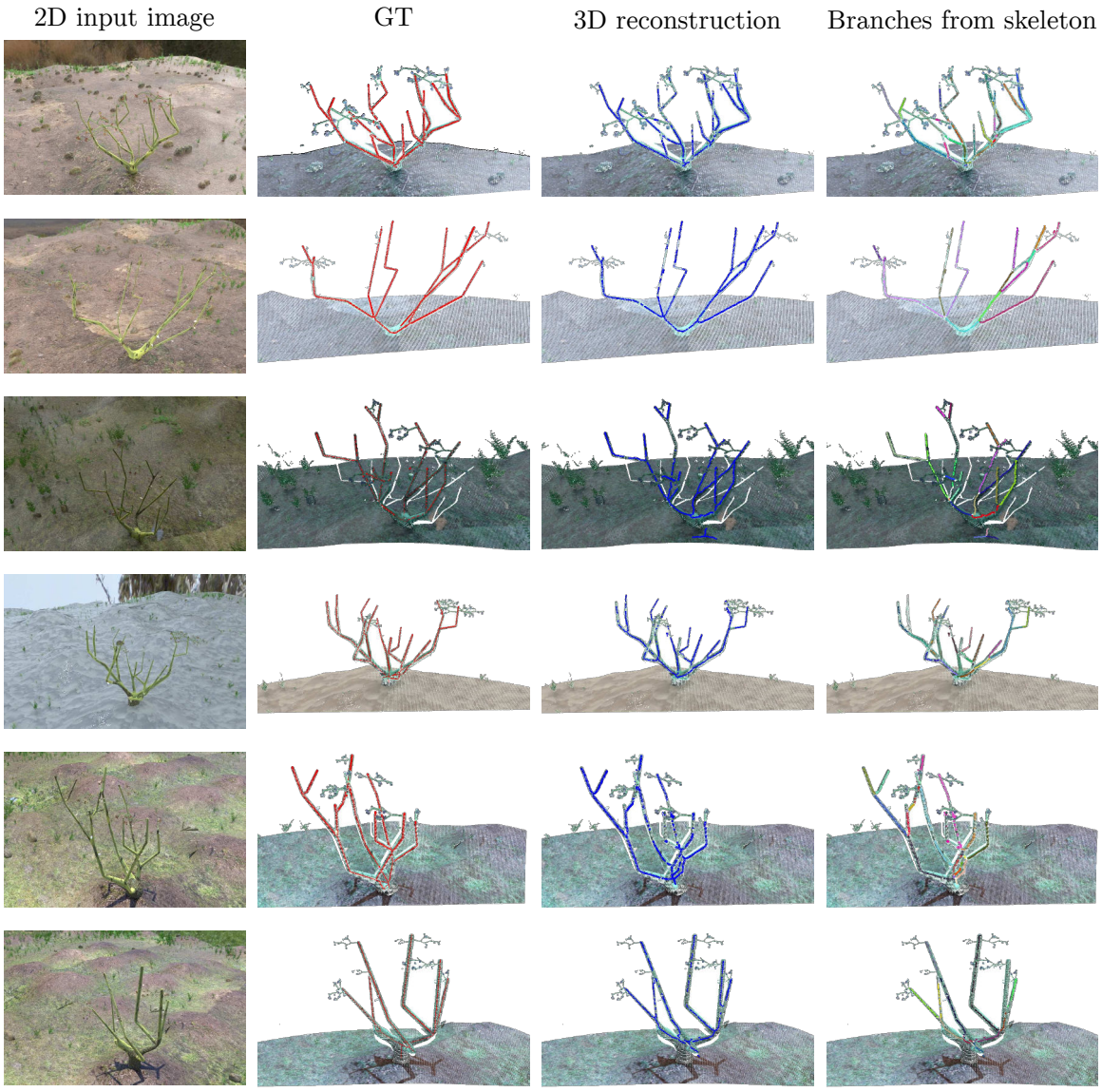


Figure 15: Output examples of the 3D skeletons obtained by our method (blue) and the ground truth skeleton (red), both superimposed over the pointcloud of the plant. The branches found using Algorithm 1 are also shown in the third column (each branch is marked using different colors).

6. Conclusions and Future Work

A new method for the segmentation and 3D reconstruction of rose bushes from stereo images was presented. This method is part of a robotic system that is capable of moving through a garden towards a rose bush and pruning it according to a series of rules. The proposed vision method tries to solve this task without making assumptions about the characteristics of the plant, the type of environment, or the lighting conditions. The method is divided into several steps that try to improve the robustness of the result and to gather all the necessary information that allows the robot to select the branches to be pruned.

The branch segmentation step is the most important of the whole process, since it allows to select only those parts of the image that are branches (differentiating them from the general background, as well as other surrounding elements that can be very similar to branches, such as sticks, support stakes, fences, etc.) so that the rest of the processing pipeline can work with correct data. For this reason, the proposed method was specifically adjusted to work well in this type of task.

Five different rose bushes datasets were used to evaluate the pipeline, three of them compiled by the authors and two from the state of the art, including interior and exterior environments, as well as different types of rose bushes, backgrounds, and lighting conditions. The segmentation method obtained an average F_1 score of 77 % at the pixel level. It is 8.18 % better than the next best result from the state of art. When evaluating at the branch level, the method correctly detected 88 % of the branches, 7.33 % better than the next best result. In addition, the significance of these results was validated by statistical tests.

The process proposed for the combination of segmentation and disparity improved the accuracy of both results, increasing the segmentation F_1 score by 1.26 %, and reducing the RMSE of the disparity calculated at the branch level from 0.2527, for the original algorithm, to 0.0869.

Five different algorithms were evaluated for the 2D skeletonization, from which Zhang & Suen [60] was chosen to create the skeleton of the segmented binary map of the plant. This method was chosen because it obtains a good F_1 score of 91.06 % and also generates fewer small branches than the other methods, which is convenient when dealing with thin objects with many bifurcations.

The 3D reconstruction of the plant is obtained by using the camera parameters, the branching search algorithm, the disparity map and the segmentation obtained by the previous steps. The overall reconstruction is on average 0.859 cm far from the ground truth, making it an accurate reconstruction both quantitatively and qualitatively. At the same time, the branches found in 2D helps the method to find the branching structure in 3D, without using any geometrical primitives or having different views of the plant.

As future work, the segmentation and disparity methods could be improved by including additional input information, such as the curvilinear structures obtained by RUSTICO. It could also be improved the calculation of the branches to be pruned, so that, in addition to the pruning rules currently considered, take into account the shape and appearance of the rose bush, so that, for example, it does not grow in an unbalanced way (more on one side than on the other).

Acknowledgments

This work was funded by the European Horizon 2020 program, under the project TrimBot2020 (grant No. 688007).

Appendix A. State-of-the-art segmentation methods

The proposed FCSN segmentation network was compared with the following three state-of-the-art segmentation methods:

- U-Net [46]: It is a CNN that was developed for biomedical image segmentation. This network uses a FCN divided into two phases: a contracting path and a expansive path. The feature activations of the contracting path are concatenated with the corresponding layers from the expansive path. The last layer uses a 1x1 convolution with a Softmax activation function to output class labels.

- SegNet [4]: It uses a FCN architecture for semantic pixel-wise segmentation, containing an encoder network, a corresponding decoder network, and a pixel-wise multiclass classification layer. The architecture of the encoder network is topologically identical to the 13 convolutional layers in the VGG16 network [52].

- DeepLabv3 [14]: It uses atrous spatial pyramid pooling to robustly segment objects at multiple scales with filters at multiple sampling rates to explicitly control the resolution at which feature responses are computed within the CNN. It also includes an image-level feature to capture longer range information and uses batch normalization to facilitate the training.

Appendix B. Skeletonization methods

The following five algorithms were evaluated to find the skeletonization method that obtained the best branch detection in the shortest runtime:

- Zhang & Suen [60]: This method makes successive passes through a blob of foreground pixels while removing pixels that do not satisfy a given condition in a 8-pixel neighborhood. The main process consists of two sub-iteration: One removes the south-east boundary points and the north-west corner points, and the other deletes the north-west boundary points and the south-east corner points.

- Parallel thinning [23]: It has a similar procedure to the previous method but using different thinning constrains. It also introduces an endpoint detection which generates thinner results than Zhang & Suen [60].

- 3D skeletonization [37]: This approach is also similar to the previous methods with the difference that it evaluates a 3D neighborhood. Here, 3D refers to 3D voxels rather than 3D point clouds. In our case, we use the disparity for the third dimension. Therefore, we build a $[n \times m \times d]$ input voxel array, where $n \times m$ is the size of the image and d the number of disparity layers, and evaluate a $[3 \times 3 \times 3]$ region to find the skeleton.

- Medial axis [9]: It is the locus of centers of circles which are maximal within the borders of a binary image. One property of the medial axis is that each point on the skeleton has a value which represents its distance to a boundary in the original object.

- RUSTICO [55]: It presents a brain-inspired inhibition mechanism built using trainable filters called B-COSFIRE [3] to delineate curvilinear structures. The inhibition mechanism introduces robustness to noise given that a gray scale image passes through two processes, an excitatory part and an inhibition component. The excitatory component is selective for bright lines on dark backgrounds, while the inhibitory counterpart is selective for wider dark lines on bright backgrounds. Although the paper focuses on a more general task, Strisciuglio et al. [55] uses it to segment branches as one of their experiments.

References

[1] Acharya, T., & Ray, A. K. (2005). *Image Processing - Principles and Applications*. New York, NY, USA: Wiley-Interscience.

[2] Alenya, G., Dellen, B., Foix, S., & Torras, C. (2013). Robotized plant probing: Leaf segmentation utilizing time-of-flight data. *IEEE Robotics & Automation Magazine*, *20*, 50–59.

[3] Azzopardi, G., Strisciuglio, N., Vento, M., & Petkov, N. (2015). Trainable cosfire filters for vessel delineation with application to retinal images. *Medical image analysis*, *19*, 46–57.

[4] Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*, 2481–2495. doi:10.1109/TPAMI.2016.2644615.

[5] Barth, R., IJsselmuiden, J., Hemming, J., & Van Henten, E. (2017). Synthetic bootstrapping of convolutional neural networks for semantic plant part segmentation. *Computers and Electronics in Agriculture*, .

[6] Barth, R., IJsselmuiden, J., Hemming, J., & Van Henten, E. (2018). Data synthesis methods for semantic segmentation in agriculture: A capsicum annuum dataset. *Computers and Electronics in Agriculture*, *144*, 284–296.

[7] Belton, D., Moncrieff, S., & Chapman, J. (2013). Processing tree point clouds using gaussian mixture models. *Proceedings of the ISPRS annals of the photogrammetry, remote sensing and spatial information sciences, Antalya, Turkey*, (pp. 11–13).

[8] Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, *13*, 281–305.

[9] Blum, H. (1967). A transformation for extracting new descriptors of shape. In W. Wathen-Dunn (Ed.), *Models for the Perception of Speech and Visual Form* (pp. 362–380). Cambridge: MIT Press.

[10] Botterill, T., Paulin, S., Green, R., Williams, S., Lin, J., Saxton, V., Mills, S., Chen, X., & Corbett-Davies, S. (2017). A robot system for pruning grape vines. *Journal of Field Robotics*, *34*, 1100–1122.

[11] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010* (pp. 177–186). Springer.

[12] Calvo-Zaragoza, J., & Gallego, A.-J. (2019). A selectional auto-encoder approach for document image binarization. *Pattern Recognition*, *86*, 37 – 47. URL: `http://www.sciencedirect.com/science/article/pii/S0031320318303091`. doi:`https://doi.org/10.1016/j.patcog.2018.08.011`.

[13] Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Return of the Devil in the Details: Delving Deep into Convolutional Nets. In *British Machine Vision Conference* (pp. 1–11). doi:`10.5244/C.28.6`.

[14] Chen, L.-C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. *CoRR*, *abs/1706.05587*.

[15] Cheng, G., & Han, J. (2016). A survey on object detection in optical remote sensing images. *{ISPRS} Journal of Photogrammetry and Remote Sensing*, *117*, 11 – 28. doi:`http://dx.doi.org/10.1016/j.isprsjprs.2016.03.014`.

[16] Cuevas-Velasquez, H., Li, N., Tylecek, R., Saval-Calvo, M., & Fisher, R. B. (2018). Hybrid multi-camera visual servoing to moving target. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1132–1137). IEEE.

[17] Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

[18] Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2* NIPS'14 (pp. 2366–2374). Cambridge, MA, USA: MIT Press.

[19] Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International journal of computer vision*, *59*, 167–181.

[20] Furgale, P., Rehder, J., & Siegwart, R. (2013). Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1280–1286). IEEE.

[21] Gélard, W., Devy, M., Herbulot, A., & Burger, P. (2017). Model-based segmentation of 3d point clouds for phenotyping sunflower plants. In *12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP 2017)* (pp. 459–467). volume 4.

[22] Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep Sparse Rectifier Neural Networks. *Journal of Machine Learning Research (JMLR) W&CP*, *15*, 315–323.

[23] Guo, Z., & Hall, R. W. (1989). Parallel thinning with two-subiteration algorithms. *Communications of the ACM*, *32*, 359–373.

[24] Gürel, C., ZADEH, M. H. G., & ERDEN, A. (2016). Development and implementation of rose stem tracing using a stereo vision camera system for rose harvesting robot. In *8th International Conference on Image Processing, Wavelet and Applications (IWW 2016)*.

[25] Gürel, C., Zadeh, M. H. G., & Erden, A. (2016). Rose stem branch point detection and cutting point location for rose harvesting robot. In *The 17th International Conference on Machine Design and Production, UMTIK 2016*.

[26] Hackenberg, J., Morhart, C., Sheppard, J., Spiecker, H., & Disney, M. (2014). Highly accurate tree models derived from terrestrial laser scan data: A method description. *Forests*, *5*, 1069–1105.

[27] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778).

[28] Hirschmuller, H. (2008). Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*, 328–341. doi:10.1109/TPAMI.2007.1166.

[29] Honegger, D., Sattler, T., & Pollefeys, M. (2017). Embedded real-time multi-baseline stereo. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 5245–5250). IEEE.

[30] Huang, H., Wu, S., Cohen-Or, D., Gong, M., Zhang, H., Li, G., & Chen, B. (2013). L1-medial skeleton of point cloud. *ACM Trans. Graph.*, *32*, 65–1.

[31] Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *JMLR W&CP*, *37*.

[32] Isokane, T., Okura, F., Ide, A., Matsushita, Y., & Yagi, Y. (2018). Probabilistic plant modeling via multi-view image-to-image translation. *arXiv preprint arXiv:1804.09404*, .

[33] Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (). Image-to-image translation with conditional adversarial networks, .

[34] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings IJCAI* (pp. 1137–1143). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. volume 2 of *IJCAI'95*. URL: http://dl.acm.org/citation.cfm?id=1643031.1643047.

[35] Konolige, K. (1998). Small vision systems: Hardware and implementation. In Y. Shirai, & S. Hirose (Eds.), *Robotics Research* (pp. 203–212). London: Springer London.

[36] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances In Neural Information Processing Systems*, (pp. 1–9). doi:http://dx.doi.org/10.1016/j.protcy.2014.09.007.

[37] Lee, T.-C., Kashyap, R. L., & Chu, C.-N. (1994). Building skeleton models via 3-d medial surface axis thinning algorithms. *CVGIP: Graphical Models and Image Processing*, *56*, 462–478.

[38] Livny, Y., Yan, F., Olson, M., Chen, B., Zhang, H., & El-Sana, J. (2010). Automatic reconstruction of tree skeletal structures from point clouds. *ACM Transactions on Graphics (TOG)*, *29*, 151.

[39] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 3431–3440). doi:10.1109/CVPR.2015.7298965.

[40] Mayer, N., Ilg, E., Häusser, P., Fischer, P., Cremers, D., Dosovitskiy, A., & Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. ArXiv:1512.02134.

[41] Orchard, M. T., & Bouman, C. A. (1991). Color quantization of images. *IEEE Transactions on Signal Processing*, *39*, 2677–2690. doi:10.1109/78.107417.

[42] Paproki, A., Sirault, X., Berry, S., Furbank, R., & Fripp, J. (2012). A novel mesh processing based technique for 3d plant analysis. *BMC plant biology*, *12*, 63.

[43] Pu, C., Li, N., Tylecek, R., & Fisher, B. (2018). Dugma: Dynamic uncertainty-based gaussian mixture alignment. In *3DV 2018*. URL: http://homepages.inf.ed.ac.uk/rbf/PAPERS/DUGMA18.pdf.

[44] Quigley, M., Conley, K., P Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Y Ng, A. (2009). Ros: an open-source robot operating system. volume 3.

[45] Raumonen, P., Kaasalainen, M., Åkerblom, M., Kaasalainen, S., Kaartinen, H., Vastaranta, M., Holopainen, M., Disney, M., & Lewis, P. (2013). Fast automatic precision tree models from terrestrial laser scanner data. *Remote Sensing*, *5*, 491–520. URL: http://www.mdpi.com/2072-4292/5/2/491. doi:10.3390/rs5020491.

[46] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*.

[47] Santos, T. T., Koenigkan, L. V., Barbedo, J. G. A., & Rodrigues, G. C. (2014). 3d plant modeling: localization, mapping and segmentation for plant phenotyping using a single hand-held camera. In *European Conference on Computer Vision* (pp. 247–263). Springer.

[48] Schönberger, J. L., Pollefeys, M., Geiger, A., & Sattler, T. (2018). Semantic visual localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 6896–6906).

[49] Shalabi, L. A., Shaaban, Z., & Kasasbeh, B. (2006). Data Mining: A Preprocessing Engine. *Journal of Computer Science*, *2*, 735–739.

[50] Shapiro, L., & Stockman, G. (2001). *Computer Vision*. Prentice Hall. URL: https://books.google.co.uk/books?id=FftDAQAAIAAJ.

[51] Simek, K., Palanivelu, R., & Barnard, K. (2016). Branching gaussian processes with applications to spatiotemporal reconstruction of 3d trees. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *European Conference on Computer Vision – ECCV 2016* (pp. 177–193). Cham: Springer International Publishing.

[52] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, *abs/1409.1556*. URL: `http://arxiv.org/abs/1409.1556`. `arXiv:1409.1556`.

[53] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.

[54] Strisciuglio, N., Azzopardi, G., & Petkov, N. (2019). Brain-inspired robust delineation operator. In L. Leal-Taixé, & S. Roth (Eds.), *Computer Vision – ECCV 2018 Workshops* (pp. 555–565). Cham: Springer International Publishing.

[55] Strisciuglio, N., Azzopardi, G., & Petkov, N. (2019). Robust inhibition-augmented operator for delineation of curvilinear structures. *Ieee transactions on image processing*, .

[56] Strisciuglio, N., Tylecek, R., Petkov, N., Bieber, P., Hemming, J., van Henten, E., Sattler, T., Pollefeys, M., Gevers, T., Brox, T., & Fisher, R. B. (2018). Trimbot2020: an outdoor robot for automatic gardening. In *50th International Symposium on Robotics*. VDE Verlag GmbH - Berlin - Offenbach. URL: `http://trimbot2020.webhosting.rug.nl/wp-content/uploads/2018/04/tb_isr.pdf`.

[57] Tabb, A., & Medeiros, H. (2017). A robotic vision system to measure tree traits. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 6005–6012). IEEE.

[58] Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *CoRR*, *abs/1212.5701*. URL: `http://arxiv.org/abs/1212.5701`. `arXiv:1212.5701`.

[59] Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014* (pp. 818–833). Cham: Springer International Publishing.

[60] Zhang, T., & Suen, C. Y. (1984). A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, *27*, 236–239.

[61] Zheng, L., Shi, D., & Zhang, J. (2010). Segmentation of green vegetation of crop canopy images based on mean shift and fisher linear discriminant. *Pattern Recognition Letters*, *31*, 920–925.

[62] Zheng, L., Zhang, J., & Wang, Q. (2009). Mean-shift-based color segmentation of images containing green vegetation. *Computers and Electronics in Agriculture*, *65*, 93–98.

[63] Zou, Q., Cao, Y., Li, Q., Mao, Q., & Wang, S. (2012). Cracktree: Automatic crack detection from pavement images. *Pattern Recognition Letters*, *33*, 227–238.